НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
Теплоенергетичний факультет
Кафедра автоматизації проектування енергетичних процесів і систем

До захисту допущено:
Завідувач кафедри
_____Олександр КОВАЛЬ
«___»_____2020 р.

Дипломна робота

на здобуття ступеня бакалавра
за освітньо-професійною програмою «Інформаційні технології моніторингу довкілля»
спеціальності 122 «Комп'ютерні науки та інформаційні технології»
на тему: «Програмні засоби інтелектуально-статистичної обробки гідроакустичних сигналів»

Виконав:
студент IV курсу, групи ТМ-62
Михалько Владислав Геннадійович

_____

Керівник:
Доцент, кандидат технічних наук
Кублій Лариса Іванівна

_____

Рецензент:
Доцент, кандидат технічних наук
Самарай Валерій Петрович _____

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

Київ – 2020 року

Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”

Факультет теплоенергетичний
Кафедра автоматизації проектування енергетичних процесів і систем
Рівень вищої освіти перший (бакалаврський)
Напрям підготовки 122 Комп’ютерні науки та інформаційні технології
Спеціалізація Інформаційні технології моніторингу довкілля

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____Олександр КОВАЛЬ
(підпис)
”___”_____2020р.

ЗАВДАННЯ

на дипломну роботу студенту

Михальку Владиславу Геннадійовичу
(прізвище, ім’я, по батькові)

1. Тема роботи Програмні засоби інтелектуально-статистичної обробки гідроакустичних сигналів

керівник роботи Кублій Лариса Іванівна, к.т.н., доцент
(прізвище, ім’я, по батькові науковий ступінь, вчене звання)

затверджена наказом вищого навчального закладу від “25” травня 2020 р. № **1168-с**

2. Строк подання студентом роботи 14 червня 2020 р.

3. Вихідні дані до роботи Python, PostgreSQL, scikit-learn, docker

4.Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) Створити графічний користувацький інтерфейс, створити модель машинного навчання для дослідження гідроакустичних сигналів, інтегрувати модель з застосунком

5. Перелік ілюстративного матеріалу
Структура датасету, засоби розробки, модель машинного навчання, структура програмного продукту, архітектура системи, схема бази даних, карта взаємодії з графічним інтерфейсом, приклади роботи

6. Публікації: Матеріали VII Науково-практичної дистанційної конференції фахівців з розробки програмного забезпечення (VII Scientific and Practical Virtual Conference of Software Development Specialists) 2020 року, м. Київ, КПІ ім. Ігоря Сікорського, 1 липня 2020 року.

7. Дата видачі завдання ”15” __жовтня__2019_ р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітки
1.	Затвердження теми роботи	03.12.19	
2.	Вивчення та аналіз задачі	13-19.04.20	
3.	Розробка архітектури та загальної структури системи	20-26.04.20	
4.	Розробка структур окремих підсистем	27.04-03.05.20	
5.	Програмна реалізація системи	04-13.05.20	
6.	Оформлення пояснювальної записки	14-16.05.20	
7.	Захист програмного продукту	17.05.20	
8.	Передзахист	09.06.20	
9.	Захист	16.06.20	

Студент _____ ____Владислав МИХАЛЬКО_____
(підпис) (прізвище та ініціали,)

Керівник роботи _____ ____Лариса КУБЛІЙ_____
(підпис) (прізвище та ініціали,)

АНОТАЦІЯ

Мета роботи — створення програмного продукту для інтелектуально-статистичної обробки гідроакустичних сигналів. Для написання системи було обрано об’єктно-орієнтовану мову програмування Python. Графічний користувацький інтерфейс системи реалізованого за допомогою пакету PyQt5. Для проведення обробки використано бібліотеки статистичного напряму Pandas, Numpy, Scipy. Для побудови інтелектуальної моделі — пакет scikit-learn. Візуалізація реалізована за допомогою бібліотеки MatPlotLib.

Записка містить 83 сторінки, 24 рисунки, 1 таблицю, 4 додатки і 21 посилання.

Ключові слова: python, машинне навчання, аналіз даних, інтерпретована мова програмування, візуалізація, статистична обробка, гідроакустика.

ABSTRACT

The purpose of the work is to implement a software product for intellectual and statistical processing of hydroacoustics signals. Object-oriented Python programming language was chosen for creation of the system. Graphical user interface implemented using package PyQt5. Pandas, Numpy, Scipy libraries were used for statistical data processing. Machine learning model implemented using scikit-learn package. Data visualization is implemented using MatPlotLib library.

The note contains 83 pages, 24 figures, 1 table, 4 attachments and 21 links.

KEYWORDS: python, machine learning, data analysis, interpreted programming language, visualization, statistical processing, hydroacoustics.

ЗМІСТ

ВСТУП

Гідроакустика вивчає явища, які відбуваються у водному середовищі і пов’язані з утворенням, розповсюдженням і прийомом акустичних хвиль. Незважаючи на досить велику кількість гідроакустичних пристроїв, у гідроакустиці неминуче виникають проблеми, пов’язані з точністю даних — отриманими сигналами.

Незважаючи на перспективність даного напряму, розвиток якого може вплинути на велику кількість галузей (таких як, наприклад, військова, розвідницька, рибопошукова тощо), на теперішній час у вільному і комерційному доступі відсутнє програмне забезпечення для виконання аналізу та обробки гідроакустичних сигналів. Якщо ж таке існує, то в більшості випадків воно або працює тільки зі спеціальним апаратним забезпеченням і не може бути використане без нього, або є нерозповсюджуваним за межі організацій.

Метою роботи є створення програмного продукту для інтелектуально-статистичного аналізу та обробки гідроакустичних сигналів, яке не буде залежне від специфічного апаратного забезпечення, може бути використане як в навчальних цілях, так і для виконання досліджень у галузі гідроакустики і зможе послужити основою для написання більш точного вузьконаправленого продукту.

Завданнями роботи є:

— створення модулів для зчитування і запису даних до файлового дерева та бази даних;

— створення компонентів для обробки й аналізу гідроакустичних сигналів;

— створення моделі машинного навчання для інтелектуального аналізу гідроакустичних сигналів та інтеграція з програмним продуктом;

— створення графічного користувацького інтерфейсу для взаємодії з продуктом.

Завдяки використанню сучасних технологій і підходів до розробки програмного забезпечення програмний продукт має нескладний і зрозумілий інтерфейс, швидко працює і не залежить від операційної системи, на якій використовується.

Середовищем розробки програмної системи обрано PyCharm 2020 (від JetBrains), мовою програмування — Python. Для створення бази даних і роботи з нею — система керування реляційними базами даних PostgreSQL. Для простоти розгортання і розділення компонентів використовується контейнер Docker.

У першому розділі пояснювальної записки сформульовано постановку задачі, описано вхідні дані і компоненти системи. У другому — подано теоретичні відомості про предметну область. У третьому розділі обґрунтовано вибір засобів для створення системи. Програмну реалізацію описано в четвертому розділі. У п’ятому розділі описано процес розгортання системи і подано приклади взаємодії з нею.

1. РОЗРОБКА СИСТЕМИ ДЛЯ РОЗПІЗНАВАННЯ ТА ІНТЕЛЕКТУАЛЬНО-СТАТИСТИЧНОЇ ОБРОБКИ ГІДРОАКУСТИЧНИХ СИГНАЛІВ

Система для інтелектуально-статистичної обробки гідроакустичних сигналів і візуалізації інформації повинна вміти працювати з файловою системою, обробляти здобуту інформацію і виконувати її візуалізацію, надавати зрозумілий користувацький інтерфейс. Гідроакустичні сигнали приймаються у вигляді файлів з форматом, що може бути налаштований у вікні конфігурації. Система повинна надавати можливість отримати корисну для користувача інформацію з оброблюваних гідроакустичних сигналів. За допомогою даного продукту користувач повинен мати змогу отримати певну інформацію про водне середовище, в якому був записаний гідроакустичний сигнал. Програмний продукт повинен вміти виконувати обробку поданого гідроакустичного сигналу для перевірки наявності аномалій у ньому.

1.1 Задача створення інформаційної системи

Створювана програмна система має розв’язувати ряд задач:

— зчитування і збереження вхідної інформації про гідроакустичні сигнали, як при роботі з файловою системою, так і з базою даних;

— перетворення інформації з сирого файлу до типу, з яким можна працювати напряму засобами обраної мови програмування;

— інтелектуально-статистичну обробку вхідної інформації;

— візуалізацію вхідної та вихідної інформації.

Для розв’язання поставлених задач треба використати існуючі програмні продукти з метою спрощення процесу підготовки вхідних даних, їхньої подальшої обробки і візуалізації.

1.2 Вхідні дані

Оскільки даний продукт реалізовує, зокрема, модель машинного навчання, то вхідні дані можна розділити на дві схожі між собою категорії, які призначаються для двох різних цілей.

Перша категорія — великий набір даних, призначений для створення й підготовки моделі машинного навчання. Ці дані включають велику кількість записів з гідроакустичних пристроїв. Вони надають можливість здійснити тренування й тестування створеної моделі.

Друга категорія — це безпосередньо дані, які можна надсилати до програми, виконувати обробку й одержувати вихідну інформацію. Незважаючи на те, що дана вибірка не є обов’язковою для корекції моделі, вона може бути використана для покращення її роботи.

Вхідні дані подаються у вигляді файлів зі структурою, вказаною в таблиці 1.1.

Таблиця 1.1. Формат вхідних даних

Заголовок файла	4 байти
P	2 байти
Vx	2 байти
Vy	2 байти
Vz	2 байти

Кожен файл складається з заголовка довжиною 4 байти, який містить тільки нульові біти. Після заголовка, блоками по чотири двобайтних значення записано дані

для певного моменту часу. Ці дані включають гідроакустичний тиск (P) і проекції коливальної швидкості на відповідні осі (Vx, Vy, Vz). Часова довжина інформації файла становить 10 секунд, значення частоти дискретизації — 2048. Таким чином, кожен блок подає дискретизовану інформацію в моменти часу з перервами 1/2048 секунди. У такому вигляді дані не можуть бути використані для тренування моделі. Після їхнього зчитування необхідно виконати попередню обробку перед їхнім використанням для тренування або тестування моделі машинного навчання.

У наданому масиві всі файли розподіляються на дві частини: множина для тренування моделі та множина для її тестування. У половині всіх файлів було штучним шляхом прибрано сигнали. Після цього, кожному файлу в відповідність було встановлено булеве значення про присутність у ньому аномального сигналу. Це значення було істинним для тієї частини файлів, які не піддалися змінам, і хибне для тих, які зазнали модифікації.

1.3 Основні компоненти системи

Для розв’язання створюваною системою вказаних проблем треба розробити такі компоненти вищого рівня:

— основний додаток, який буде виконуватися в головному процесі;

— вікна візуалізації даних, які будуть виконуватися в побічних процесах;

— мережу для взаємодії контейнерів системи;

— базу даних зчитаної інформації (розміщується в контейнері).

Система з обробки гідроакустичних сигналів має надавати користувачеві можливість переглядати оброблену інформацію як візуально, так і в текстовому форматі.

Для роботи з проектом у програмному продукті має бути створено три інтерфейси, кожен з яких призначається і використовується з різною метою:

— графічний користувацький інтерфейс, розроблений за допомогою PyQt5, для простоти використання й виконання основних функцій додатку;

— інтерфейс Makefile з інтерпретатором мови і допоміжною оболонкою IPython для доступу до всіх створених програмних класів, методів і функцій системи;

— інтерфейс Jupyter для використання калькуляційних функцій додатку і збереження історії дій.

Компонент бази даних має розташовуватися всередині docker-контейнера для простоти розгортання проекту, але за умови правильної конфігурації середовища можна налаштувати запис даних у певну директорію машини.

Процес зчитування й запису даних має виконуватися всередині основного додатку, він може взаємодіяти як з окремими файлами системи, так і з базою даних. Варто зазначити, що формати збереження файлів і записів у базі даних різні. На відміну від формату файла, вказаного вище, в базі даних записи зберігаються рядками з усіма необхідними полями, але згідно зі специфікою обраної системи керування реляційними базами даних.

Модель машинного навчання створюється поза межами даної системи, але вона зберігається за допомогою вбудованої бібліотеки мови Python в бінарному форматі з розширенням .pikle. Вона зчитується й перетворюється до необхідного об’єкта при кожному запуску додатку.

Головний додаток містить модуль для зв’язування всіх компонентів між собою і налаштування взаємодії між ними.

На рисунку 1.1 подано діаграму компонентів, яка показує потік даних і зв’язок підсистем між собою.

Для взаємодії з явно виокремленим елементом бази даних, який розміщується в контейнері, треба використати локальну мережу docker-compose. Вона надає можливість легко пов’язувати контейнери між собою і надає їм з’єднання з зовнішніми застосунками. Компонент головного додатку вимагає запуску Python-інтерпретатора на локальній машині, він не буде обернений в контейнер з метою більш економного

використання ресурсів комп’ютера. Для кожного відкритого вікна візуалізації з головного додатку запускається окремий Python-процес за допомогою вбудованої бібліотеки multiprocessing.

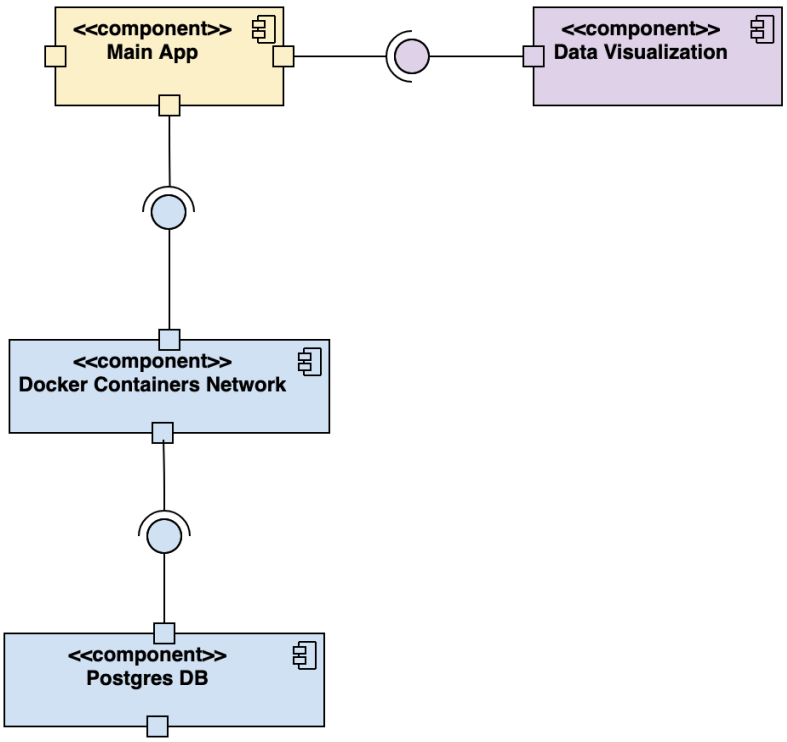


Рисунок 1.1 — Діаграма компонентів

Після закриття вікна відповідний процес зупиняється і знищується. Кількість вікон візуалізації не обмежено програмними засобами, але варто брати до уваги можливості машини, на якій запускається програмна система, і не відкривати велику кількість додаткових вікон, якщо ресурси комп’ютера не надають можливості працювати з багатьма вікнами одночасно.

1.4 Потенційні користувачі

Система для розпізнавання та інтелектуально-статистичної обробки гідроакустичних сигналів знаходиться у вільному доступі і не включає ніяких

ліцензійних обмежень, що говорить про можливість її використання з будь-якою метою, в тому числі комерційною.

Серед потенційних користувачів можна виокремити учнів навчальних закладів, які вивчають галузь гідроакустики, для поглиблення знань і виконання досліджень. Також вона може використовуватися фахівцями в галузі гідроакустики, яким необхідно виконувати певний аналіз гідроакустичних сигналів.

Програмна система є легко розширюваною, що надає можливість будувати специфічні додатки, взявши її за основу.

2. АНАЛІЗ СУЧАСНОЇ ГІДРОАКУСТИКИ І ЇЇ ІНТЕГРАЦІЯ З МАШИННИМ НАВЧАННЯМ

Гідроакустика — наука про явища, які відбуваються в водному середовищі і пов’язані з випромінюванням, розповсюдженням і прийомом акустичних хвиль [1]. Вона включає в себе питання розробки і створення гідроакустичних засобів, призначених для використання в водному середовищі.

Гідроакустика досить молода наука, яка швидко розвивається в даний час і яка має перспективи росту. Її виникненню передує довгий шлях розвитку теоретичної і прикладної акустики. Але в гідроакустиці існує велика кількість нерозв’язаних проблем.

Сучасні гідроакустичні сенсори вже досягли видимого на сьогоднішній день ліміту і покращити результуючі дані, які надходять з встановлених сенсорів, на даний момент не є можливим. Одним з розв’язань даної проблеми є розробка програмних засобів для покращення обробки й аналізу можливих для збору даних.

Машинне навчання, як і гідроакустика, є молодою наукою, але у зв’язку з технологічним розвитком і популярністю комп’ютерних наук у даний час цей напрямок вже має велику кількість шляхів для розв’язання проблем у багатьох сферах.

2.1 Засоби збору точнішої гідроакустичної інформації

Відомо, що збільшення ефективності акустичних систем різного призначення пов’язане з покращенням співвідношення сигнал/шум. Це призводить, зокрема, до потреби більш повного використання інформації, яку можна одержати в акустичному полі.

Отримати додаткову інформацію про акустичне поле при заданих обсягах можливо шляхом щільної та злитної (по простору) упаковки приймача звукового тиску або шляхом встановлення в вибіркові точки середовища (в яких розташовані приймачі) приймачів першого і більш високих порядків.

Введення в практику акустичних вимірів поряд з приймачами акустичного тиску приймачів більш високих порядків надає можливість визначати градієнти звукових тисків, що в свою чергу надає можливість на принципово новій основі ставити і розв'язувати актуальні задачі сучасної гідроакустики. Це призвело до виникнення комбінованих прийомних систем з векторним приймачем.

Переваги комбінованих прийомних систем з векторним приймачем [2], порівняно з традиційними, побудованими на приймачах акустичного тиску, зростають по мірі зменшення кількості прийомних модулів і збільшення області простору їхнього розміщення в середовищі.

2.2 Загальні відомості про характеристики акустичних полів

Як відомо, акустичне поле коливань виникає і розповсюджується в пружних матеріальних середовищах, які оточують досліджувані об'єкти (повітря, вода, ґрунт, стіни будівель тощо), у вигляді акустичних хвиль.

Акустичні хвилі — це механічні коливання, які розповсюджуються в матеріальному (газоподібному, рідкому або твердому) середовищі.

У широкому розумінні звук — це акустичні коливання матеріального середовища, або акустичні хвилі, які сприймаються слухом людини. Але на практиці терміни звукові хвилі та акустичні хвилі часто прирівнюють.

Теорія звуку в її класичній формі будується на основі законів руху рідини і газу з врахуванням ряду особливостей коливальних рухів з малою амплітудою. У свою чергу рух рідини і газу підпорядковується законам гідро- та аеродинаміки. Оскільки рівняння

гідро- й аеродинаміки з математичної точки зору записуються однаково, під рідиною в цьому випадку також можна розуміти й газ.

У загальному випадку рівняння гідродинаміки є нелінійними і складно піддаються розв'язанню. Тому часто спрощують припущення, які дають змогу розв'язати задачу з тою чи іншою мірою наближення.

Так, при розв'язанні задач, пов'язаних з рухом рідин і газів, гарну апроксимацію надає гідродинаміка ідеальної рідини, позбавленої в'язкості. Без врахування в'язкості можна розрахувати підйомну силу крила літака. Однак при цьому не можна обчислити опір газу або рідини до руху тіла.

Якщо досліджувані швидкості значно менші від швидкості звуку в даному середовищі, то можна знехтувати стискуваністю рідин і газів. Гідродинаміка нестискуваної рідини в певних межах застосовна до розв'язання задач не тільки для рідин, які мало стискаються, але й для газів, хоч на перший погляд здається незрозумілим, як можна вважати газ нестискуваним. Передбачувана нестискуваність є наслідком того, що при швидкостях, менших за швидкість звуку, будь-які виміри тиску, викликані рухом тіла, розповсюджуються у вигляді хвилі зі швидкістю звуку і випереджають саме тіло, яке рухається. У результаті, коли різні частини тіла переходять у нові області середовища, вони застають там тиск, який вже змінився під дією руху тіла в попередні моменти часу. При обмежених розмірах тіла, яке рухається, деформація газу залишається незмінною, що еквівалентно газу, який не стискається.

При швидкостях, які наближаються до швидкості звуку або перевищують її, рідини й гази вже не можна вважати середовищами з незмінним тиском, крім того, доводиться враховувати особливі значення явищ теплопровідності при процесах, які швидко відбуваються.

При строгому розв'язанні задачі про коливальні рухи в суцільних середовищах рідин необхідно враховувати стискуваність газів і рідин.

При таких рухах в тілах, які мають досить велику протяжність, виникають явища, які називають хвилями. Хвилі з кінцевою швидкістю (швидкістю звуку) передають

виникаючі деформації і тиски у всі боки від місця їхнього виникнення. Кожне середовище залежно від величини своєї стискуваності і щільності характеризуються певною швидкістю звуку.

Основні питання акустики розв’язуються в передбачено малих амплітудах коливання з врахуванням стискуваності середовища.

2.3 Машинне навчання

Машинне навчання полягає в добуванні знань з даних [3]. Це наукова галузь на перетині статистики, штучного інтелекту і комп’ютерних наук. Вона також відома як прогнозна аналітика або статистичне навчання.

В останні роки застосування методів машинного навчання в повсякденному житті стало буденним явищем. Багато сучасних веб-сайтів і пристроїв використовують алгоритми машинного навчання, починаючи з автоматично створених рекомендацій щодо перегляду фільмів, замовлення їжі, покупки продуктів, і закінчуючи персоналізованими онлайн-радіотрансляціями та розпізнаванням друзів на фотографіях. Вийшовши за межі комерційних додатків, машинне навчання вже справило величезний вплив на наукові дослідження, які керуються даними. Інструменти машинного навчання використовують для розв’язання різних наукових задач (дослідження зірок, пошук віддалених планет, відкриття нових часток, аналіз послідовності ДНК, а також розробка персоналізованих методів лікування раку). Незважаючи на те, що машинне навчання як напрям з’явилося не так давно, можливості його застосування до будь-якої галузі надали цій науці швидкого розвитку порівняно з іншими галузями.

Машинне навчання має велику кількість напрямів, які використовуються для передбачення, поділу на групи, виявлення аномалій тощо. Такий широкий вибір надає можливість використовувати машинне навчання майже для будь-якої галузі досліджень. За умови правильного підбору методів і коректного тренування,

результуюча модель часто може перевершити вміння людини при розв’язуванні специфічних задач.

У наш час машинне навчання також використовують при розв’язуванні обчислювальних задач, для яких розробка і програмування явних алгоритмів є досить складним або навіть неможливим процесом.

2.4 Огляд методів машинного навчання

Незважаючи на великі можливості машинного навчання, для кожної галузі, де його використовують, необхідно обирати правильні методи для розв’язання проблем. Для сфери гідроакустики можуть бути використані такі напрями, як кластеризація, класифікація і виявлення аномалій.

Кластерний аналіз [4] — задача розбиття заданої вибірки об’єктів (ситуацій) на неперетинні множини, які називають кластерам, так, щоб кожен кластер складався зі схожих об’єктів, а об’єкти різних кластерів істотно відрізнялися.

Задача кластеризації відноситься до широкого класу задач навчання без вчителя, який також називають unsupervised learning.

Вхідні дані для задачі кластеризації поділяють на два підкласи: ознаковий опис і матриця відстаней між об’єктами.

При ознаковому описі об’єктів кожен об’єкт описується набором своїх характеристик, які називають ознаками. Ознаки можуть бути як числовими так і нечисловими.

При використанні матриці відстаней між об’єктами кожен об’єкт описується відстанями до всіх інших об’єктів навчальної вибірки. Матриця відстаней може бути обчислена за матрицею ознакових описів об’єктів нескінченною кількістю шляхів, залежно від того, як ввести функцію відстані (метрику) між ознаковими описами. Часто використовується евклідова метрика, проте цей вибір у більшості випадків є евристикою і обумовлений лише міркуваннями зручності.

Зворотне завдання — відновлення ознакових описів за матрицею попарних відстаней між об’єктами в загальному випадку не має розв’язання, а наближений розв’язок не є єдиним і може мати істотну похибку. Це завдання розв’язується методами багатовимірного шкалювання.

Таким чином, постановка задачі кластеризації за матрицею відстаней є більш загальною. З іншого боку, при наявності ознакових описів часто вдається будувати більш ефективні методи кластеризації.

Кластеризація ставить перед собою такі цілі, як розуміння даних шляхом виявлення кластерної структури (розбиття вибірки на групи схожих об’єктів дає можливість спростити подальшу обробку даних і прийняття рішень, застосовуючи до кожного кластеру свій метод аналізу, тобто стратегію “розділяй і володарюй”), стиснення даних (якщо вхідна вибірка надлишково велика, то можна скоротити її, залишивши по одному найбільш типовому представникові від кожного кластера), виявлення новизни (виділяються нетипові об’єкти, які не вдається приєднати до жодного з кластерів).

У першому випадку кількість кластерів намагаються зробити найменшою. У другому випадку важливіше забезпечити високий (або фіксований) степінь подібності об’єктів всередині кожного кластера, а кластерів може бути будь-яка кількість. У третьому випадку найбільший інтерес становлять окремі об’єкти (їх називають ізольованими), які не вписуються в жоден з кластерів.

У всіх цих випадках може застосовуватися ієрархічна дивізимна кластеризація, коли великі кластери дробляться на більш дрібні, а ті в свою чергу, дробляться на ще дрібніші тощо. Такі завдання називаються завданнями таксономії.

Результатом таксономії є деревоподібна ієрархічна структура. При цьому кожен об’єкт характеризується перерахуванням всіх кластерів, до яких він належить, в порядку від великого до дрібного. Візуально таксономія подається у вигляді графіка, який називають дендограмою.

До методів кластеризації, які найчастіше застосовуються в машинному навчанні, належать такі:

— ієрархічні. Ієрархічні алгоритми можуть бути агломеративними і дивізимними. В агломеративних (об'єднуючих) алгоритмах вважається, що спочатку кожен об'єкт утворює окремий кластер; надалі об'єкти об'єднуються і таким чином формуються крупніші кластери. В ієрархічних дивізимних методах сукупність об'єктів, яка спочатку вважається одним кластером, поступово ділить на дрібніші кластери; формування кластерів припиняють тоді, коли значення міри подібності перевищить задану величину або коли буде утворено задану кількість кластерів;

— K-середніх — дає можливість поділити сукупність об'єктів на k кластерів, так щоб кожен об'єкт належав до кластера з найближчим до нього середнім значенням;

— C-середніх — удосконалений метод K-середніх, в якому для кожного об'єкта сукупності розраховується міра його належності кожному з кластерів; метод розбиває сукупність на нечіткі підмножини;

— виділення пов'язаних компонент — виявляє відносно компактні гіперсферичні кластери.

Ще один розділ машинного навчання — класифікація [5]. Даний метод розв'язує таку задачу. Є множина об'єктів або ситуацій, розподілених за певними ознаками на класи. Також задана кінцева множина об'єктів, для яких вказано, до яких класів вони належать, ця множина називається тренувальною вибіркою. Класова належність інших об'єктів невідома. Необхідно побудувати алгоритм, здатний класифікувати об'єкт з першої множини. Класифікувати об'єкт — значить вказати номер або найменування класу, до якого він належить. Класифікація об'єкта — номер або найменування класу, який видає алгоритм класифікації, в результаті його використання на заданому об'єкті. В математичній статистиці задачі класифікації також називають задачами дискримінантного аналізу. У машинному навчанні задача класифікації об'єктів відноситься до розділу навчання з вчителем, або supervised learning.

При використанні методу класифікації може утворюватися різна кількість класів і класи можуть мати різні властивості:

— двокласова класифікація — найпростіший у технічному відношенні випадок, який є основою для розв’язання більш складних задач;

— багатокласова класифікація — кількість класів досягає багатьох тисяч (наприклад, при розпізнаванні ієрогліфів), задача класифікації стає значно складнішою;

— неперетинні класи — об’єкти чітко відносяться до певних класів;

— перетинні класи — об’єкт може одночасно належати до багатьох класів;

— нечіткі класи — треба визначати ступінь належності об’єкта до кожного з класів, як правило, це задається значенням від 0 до 1.

Метод класифікації використовують при розв’язуванні великої кількості прикладних задач, зокрема: задачі медичної діагностики, передбачення родовищ корисних копалин, оптичного розпізнавання символів тощо.

Виявлення аномалій — це процес знаходження неочікуваних елементів або подій в множині даних, які значно відрізняються від норми. Виявлення аномалій частіше застосовують до немаркованих даних, але це не є обов’язковим правилом.

Аномалії можуть бути трьох типів: точкові, контекстуальні й колективні.

Точкова аномалія характеризується єдиним екземпляром значення, що виділяється на фоні очікуваних. Контекстні аномалії, або як їх ще називають — умовні, — це аномалії, які проявляються тільки за певних умов. Колективні аномалії — група взаємопов’язаних значень, які виділяються на загальному фоні. Варто зазначити, що окремо кожне значення цієї категорії не являє собою аномалію, як аномалія розглядається тільки їхнє сумісне виникнення.

Існують алгоритми машинного навчання, створені спеціально для виявлення аномалій, але аномалії також можуть бути знайдені внаслідок виконання певних типів класифікації чи кластерного розподілу. Саме такий підхід використано в даній роботі.

3. ЗАСОБИ РОЗРОБКИ

Для реалізації програмного продукту було використано ряд пограмних засобів.

Середовищем розробки вибрано програмне середовище Jetbrains PyCharm 2020. Для контролю версій використано систему Git [6] і хмарне сховище gitlab з приватною репозиторією. Основною мовою для реалізації проекту вибрано мову програмування Python. Для розгортання компонентів середовища використано програмний засіб Docker [7] для контейнеризації і Docker-Compose для налагодження взаємодії між контейнерами високого рівня. Для збереження даних у табличному форматі використано систему керування реляційними базами даних PostgreSQL [8]. Також використано бібліотеки мови програмування Python такі, як: Numpy, Scipy, Pandas, Scikit-learn, Sqlalchemy, Psycopg2, Jupyter, Flake8, PyQt5, Pylint, Pytest.

3.1 Мова програмування Python

Мова програмування Python [9] — мова високого рівня загального призначення, орієнтована на підвищення продуктивності розробника і підвищення читабельності програмного коду. Синтаксис даної мови досить мінімалістичний.

Стандартна бібліотека мови Python зберігає велику кількість корисних класів, методів і функцій.

Мова Python підтримує структурне, об’єктно-орієнтоване, функціональне й аспектно-орієнтоване програмування. Основні архітектурні риси даної мови — динамічна типізація, автоматичне керування пам’яттю, абсолютна інтроспекція, механізм обробки винятків, підтримка багатопотокових обчислень, високорівневі

структури даних. Підтримується розбиття програми на модулі, які в свою чергу, можуть бути об’єднанні в пакет.

Інтерпретатор CPython [10] — еталонна реалізація, яка підтримує більшість активних платформ, у тому числі Linux-системи (Debian, Ubuntu, Arch, Kali, CentOS, RedHat), Windows і OSX. Дана мова розповсюджується під вільною ліцензією Python Software Foundation License, яка дає можливість використовувати її без обмежень в будь-яких застосунках, у тому числі в комерційних.

Також існують реалізації під інтерпретатор JVM з можливістю компіляції, CLR, LLVM та інші незалежні реалізації. Проект PyPy використовує Just-in-Time-компіляцію, яка значно збільшує швидкість виконання Python-застосунків. Мова програмування Python розвивається, нові версії з додаванням властивостей реалізуються приблизно один раз на два з половиною роки.

Мова Python не стандартизована, але роль стандарту де-факто грає CPython, яка розробляється під прямим контролем автора мови. На даний момент Python займає одну з найвищих позицій у рейтингу мов програмування згідно з TIOBE [11].

3.2 Бібліотека Numpy

Бібліотека NumPy — основний пакет наукових обчислень для Python. Переважно він створений з використанням C-розширень, що надає змогу значно пришвидшити обчислення при використанні даного пакету. Пакет містить модулі для швидкої роботи з N-вимірними масивами, складні функції, методи лінійної алгебри, швидке перетворення Фур’є, генерацію випадкових чисел.

Крім очевидних наукових застосувань, пакет NumPy також може бути використаний як ефективний багатовимірний контейнер із загальними даними. Можна визначити довільні типи даних. Це дає змогу NumPy легко й швидко інтегруватися з найрізноманітнішими базами даних.

3.3 Бібліотека SciPy

Бібліотека SciPy являє собою екосистему, яка базується на Python, з відкритим кодом програмного забезпечення для математики, науки й техніки. Даний пакет містить корисний функціонал для апроксимації, інтерполяції, екстраполяції функцій. У нього також вбудовано чимало юнітів, які допомагають справитися з numeric-based проектами.

3.4 Бібліотека Pandas

Бібліотека Pandas — це пакет для мови Python, який забезпечує швидку, гнучку і виразну структуру даних (яка називається DataFrame), розроблену для спрощення роботи зі структурованими (табличними, багатовимірними, потенційно неоднорідними) даними. Даний пакет є важливою складовою для практичного аналізу даних у реальному світі за допомогою мови програмування Python.

3.5 Бібліотека Scikit-learn

Бібліотека Scikit-learn — безкоштовна бібліотека машинного навчання для мови програмування Python. У даному пакеті представлені різні алгоритми класифікації, регресії і кластеризації, включаючи support vector machines, random-forests, k-tools і DBSCAN. Пакет оснований на взаємодії з числовими й науковими бібліотеками Python NumPy та SciPy.

3.6 Бібліотека SQLAlchemy

Бібліотека SQLAlchemy — інструментарій мови Python для роботи з SQL [12] та об’єкто-релаційним представленням, який надає розробникам програмних продуктів

повну потужність та гнучкість SQL. Бібліотека SQLAlchemy забезпечує повний набір добре відомих моделей стійкості транзакцій, розроблених для ефективного й високоефективного доступу до бази даних. Бібліотеку адаптовано до простої взаємодії з базами даних за допомогою мови програмування Python.

3.7 Бібліотека PyQt5

Бібліотека PyQt5 [13] є аналогом бібліотеки для компільованої мови високого рівня C++, яка завдяки популярності була реалізована для мови Python з перенесенням всіх можливостей оригінальної бібліотеки. Вона надає можливість створювати графічні інтерфейси з RUI [14] — багатого користувацького інтерфейсу. Пакет побудовано з використанням парадигми об’єкто-орієнтованого програмування [15]. Крім надання можливостей широкої стилізації графічних компонентів, даний пакет також підтримує систему сигналів, що дає можливість легкого налагодження зв’язку між створеними компонентами, побудованого на механізмі керування подіями.

3.8 Пакет PyLint

Пакет PyLint призначений для слідкування за станом коду проекту. Даний пакет легко конфігурується за допомогою додавання спеціального файла до головної директорії проекту. У даному файлі можна описати спеціальні правила написання коду, за дотриманням яких буде слідкувати даний пакет.

3.9 Пакет PyTest

Пакет PyTest містить досить велику кількість спеціально побудованих компонентів для покриття проекту тестами, в тому числі спеціальні засоби для підготовки середовища до виконання тестів, написання так званих “моків” [16] тощо. У

цьому пакеті створено модель перебудови структури коду для надання можливості використання спеціальних конструкцій мови для написання легко читаних тестів.

3.10 Пакет Jupyter

Пакет Jupyter Notebook — це веб-додаток з відкритим вихідним кодом, який надає можливість створювати і передавати документи з “живим” кодом, виразами, рівняннями, візуалізацією, коментарями. Його використовують для очищення, трансформації та візуалізації даних, статистичного моделювання, перевірки моделей машинного навчання тощо.

4. ОСОБЛИВОСТІ ПРОГРАМНОЇ РЕАЛІЗАЦІЇ СИСТЕМИ

Комбінація великої кількості частин програмної системи вимагала якісно підібраної архітектури з чітким розбиттям на компоненти, зв’язаність між якими буде побудована на затверджених інтерфейсах, для надання можливості легкої заміни компонентів. Всередині програмованих компонентів планувався тісний зв’язок всередині Python-модулів і слабкий зв’язок між ними, так званий підхід «Low coupling & High cohesion» [17].

Компонент бази даних огорнений в контейнер, зв’язок з ним підтримується завдяки об’єктно-реляційній системі, що надає можливість легкої заміни бази даних шляхом розгортання нового контейнеру, внесення змін в програмований компонент не потрібне.

Робота з файловими шляхами виконується через спеціальний вбудований в мову програмування пакет Path [18], що надає можливість взаємодії з файлами в будь-якій операційній системі, яка підтримує інтерпретатор CPython.

Інтелектуальний модуль є так званою обгорткою над моделлю машинного навчання для уніфікації інтерфейсу взаємодії з нею. Сама модель створена окремо від системи. Модель збережена в двійковому коді зі спеціальним форматом .pickle [19], що надає можливість легко зчитати її всередині будь-якої Python-програми. Завдяки створенню обгортки над моделлю, система має можливість її легкої заміни, незалежно від методу.

Програмований компонент головної програми побудований з використанням інтерфейсів взаємодії між модулями та класами для надання можливості легкого розширення та модифікації системи.

Графічний інтерфейс і візуалізація чітко розділені між собою, що надає можливість внесення змін в будь-яку з цих двох частин без необхідності корегування іншої.

4.1 Модель машинного навчання

Як модель машинного навчання для програмного продукту було обрано “Decision Tree”. Дана модель є одним з алгоритмів методу класифікації, але в розробленій системі її було використано для виявлення аномалій. Модель повертає булеве значення у відповідь на надісланий вектор даних. Значення “істина” — для сигналу, де виявлено аномалію, і, відповідно, значення “хибність” — для сигналу, де аномалій не було знайдено. Таким чином модель розділяє всі записи на два класи: сигнал з аномалією і сигнал без аномалії.

Для відправлення даних сигналу до моделі, сигнал перетворюється в вектор значень, який представляє спектр частот межах [1; 256]. Значення для кожної частоти було дискретизоване, а саме, спектр був перетворений в в вектор з 256 значень. Для кожного значення частоти бралося найбільше значення в околі 0,5, тобто, наприклад, для значення частоти 3 вибиралося найбільше значення функції від частоти в межах [2,5; 3,5), і так для кожної частоти в векторі.

Для створення моделі вхідний масив даних було поділено на дві рівні частини. Одна частина залишилася незмінною, а в іншій сигнал був штучно анульований. Так, для всіх незмінних сигналів була присвоєна мітка “Істина”, тобто аномалія присутня. Для всіх сигналів, які були штучно модифіковані присвоєна мітка “Хибність”, тобто аномалію не знайдено.

Після виконання такої обробки вхідної множини даних було випадково відібрано 2000 з 7370 сигналів для тренування моделі машинного навчання, всі інші сигнали були використані для тестування моделі. Опираючись на результати тестування, модель показала точність виявлення аномалій на рівні 94%.

4.2 Архітектура системи

Система побудована у вигляді зв’язаного набору компонентів вихідного коду, кожен з яких поєднаний з необхідними для нього іншими компонентами шляхом використання інтерфейсів взаємодії. Структуру програмного продукту подано на рисунку 4.1.

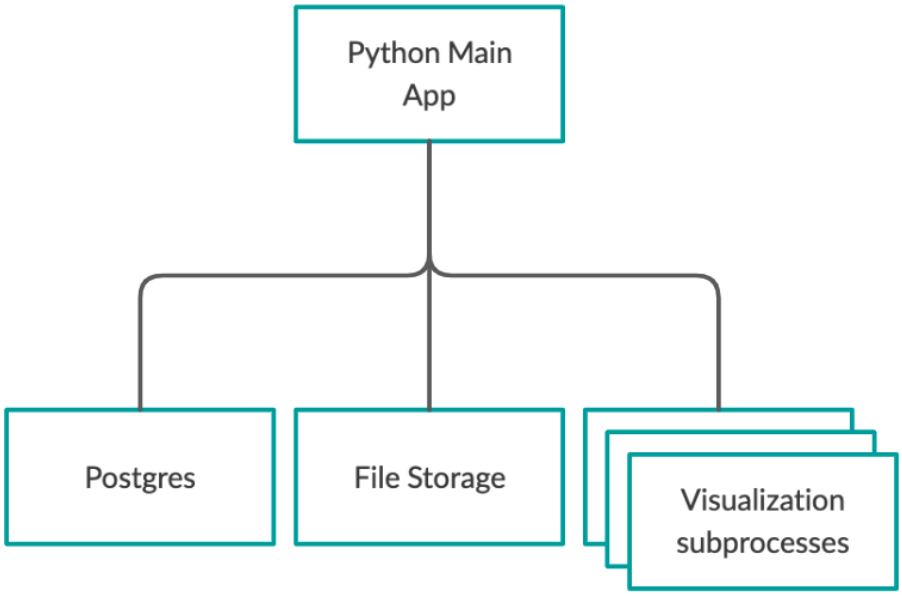


Рисунок 4.1 — Структура проекту

Даний програмний продукт призначений для одночасного використання однією людиною. Вхідними інтерфейсами для користувача виступають три компоненти — це графічний користувацький інтерфейс, створений за допомогою PyQt5, користувацький Makefile-інтерфейс та інтерфейс Jupyter-Notebook. Останні два надають можливість більш гнучкої роботи з розробленим застосунком і допоміжними засобами. В інтерфейсі Makefile створено корисні команди для взаємодії з проектом, а також додана можливість запуску інтерактивного терміналу, який обгортається в оболонку IPython, що не тільки надає можливість виконувати всі операції, які синтаксично

дозволені в мові, але й одночасно здійснювати операції з оболонкою bash такі, як зміна основного потоку введення/виведення, взаємодія з файловою системою, запуск в паралельне виконання додаткових програмних компонентів тощо.

Інтерфейс Jupyter надає можливість виконувати будь-які Python-правильні операції за допомогою веб-інтерфейсу і зберігати історію до спеціальних файлів для надання можливості їхнього подальшого відновлення і використання.

Архітектуру проекту подано на рисунку 4.2. Компоненти користувацьких інтерфейсів взаємодіють з центральним компонентом, який включає в себе налаштування взаємодії між всіма компонентами системи. Програмний компонент Data I/O включає в себе два адаптери: для роботи з файловою системою та базою даних.

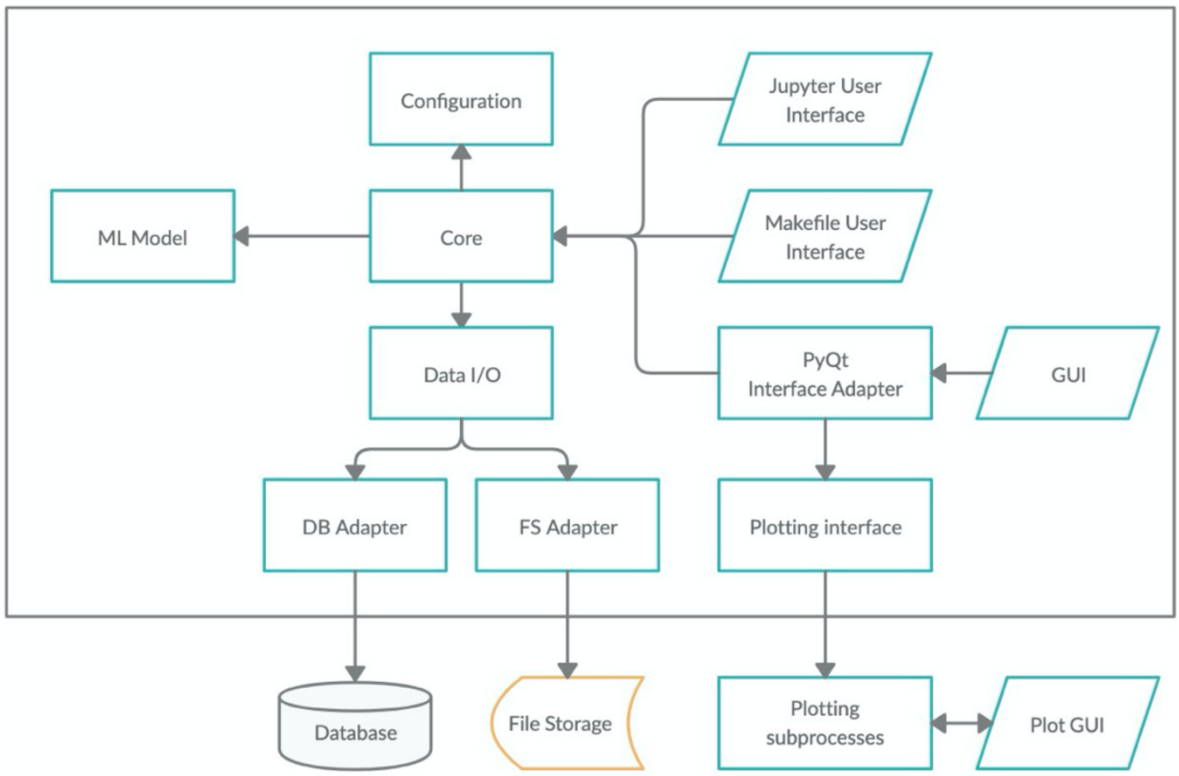


Рисунок 4.2 — Архітектура проекту

Компонент ML Model є обгорткою моделі машинного навчання для програмного продукту. Сама модель машинного навчання створена окремо, вона завантажується під

час кожного запуску додатку. За допомогою центрального компонента дані передаються між частинами програми.

За допомогою Makefile-інтерфейсу користувач отримує дещо ширші можливості. До вказаних вище можливостей додається можливість більш чіткого конфігурування системи, корегування моделі, а також більш детальна візуалізація даних (завдяки інтерактивному терміналу Python надається можливість втручатися в процес візуалізації даних і вносити свої правки в відображення оригінальної та обробленої інформації).

На діаграмі прецедентів (рисунок 4.3) вказано можливості роботи користувача.

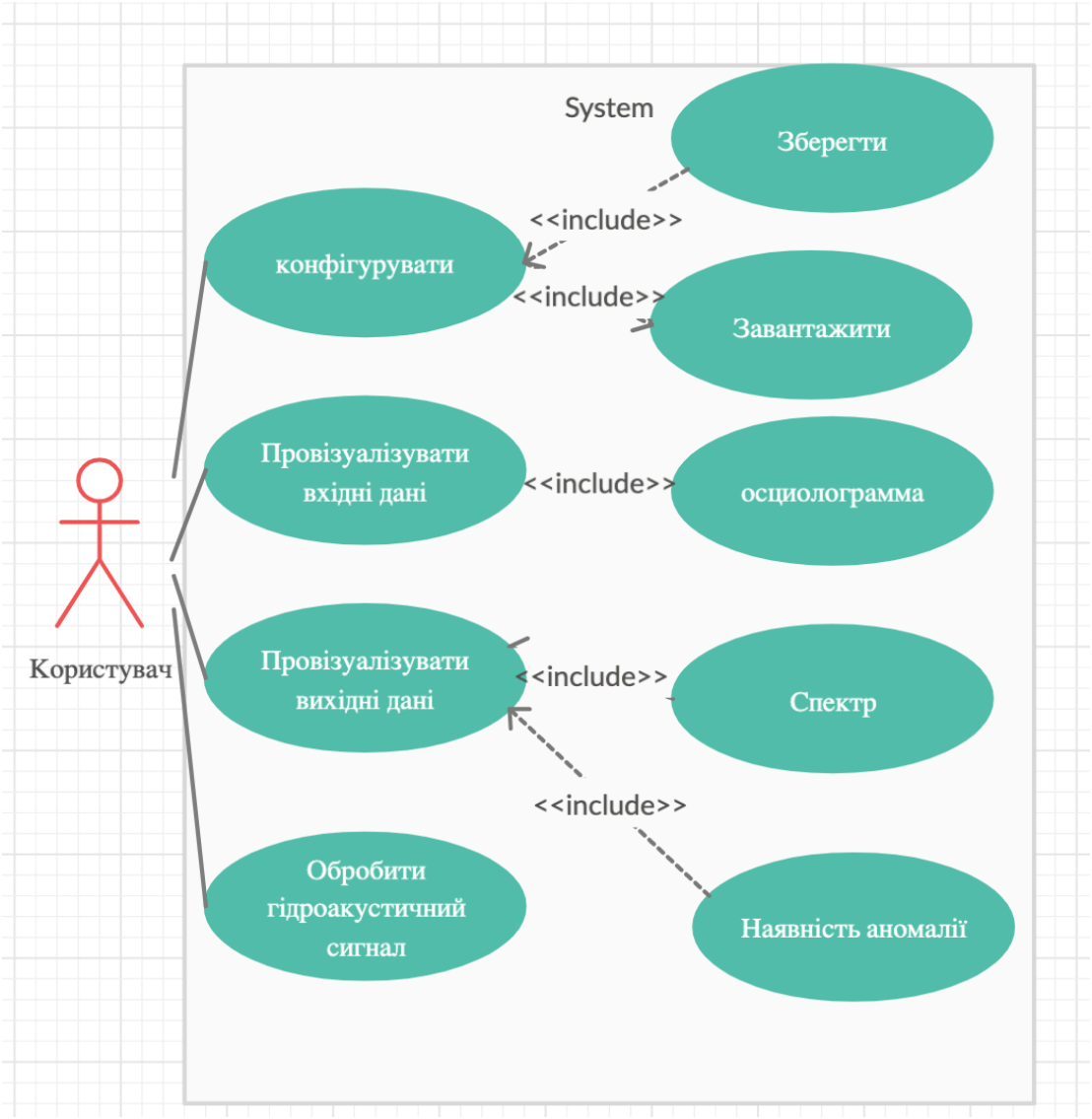


Рисунок 4.3 — Діаграма прецедентів

На рисунку 4.4 подано UML-діаграму [19] до модуля взаємодії з базою даних, який обгорнуто у власні класи для уніфікації інтерфейсу взаємодії між модулями.

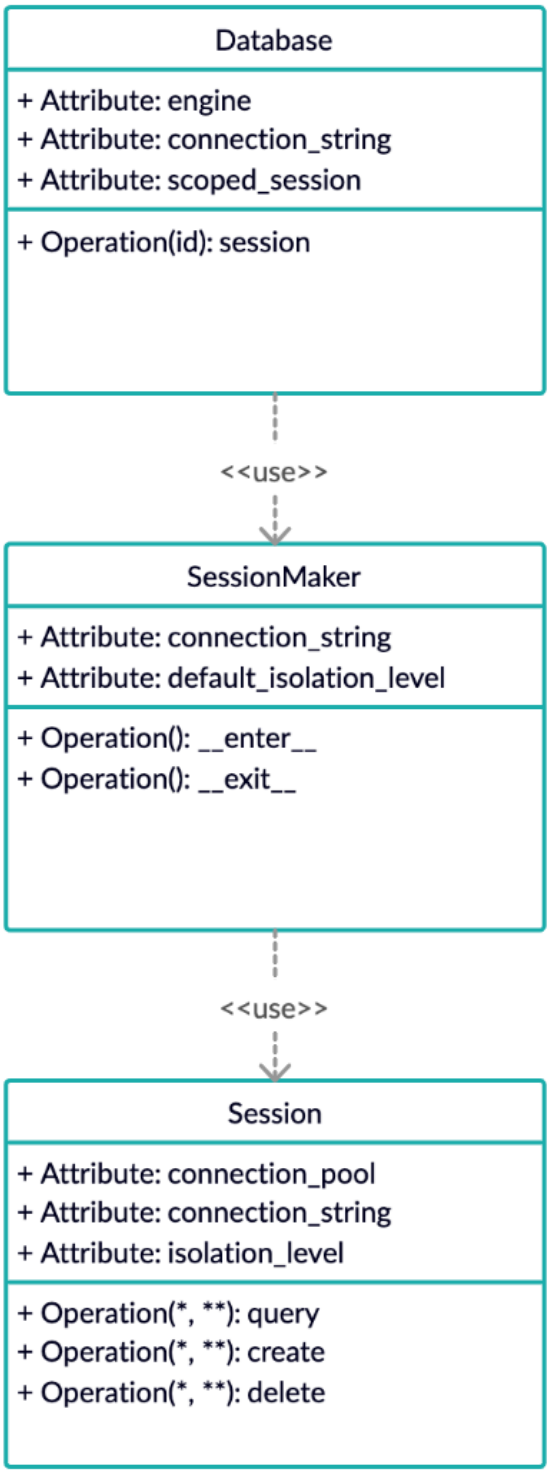


Рисунок 4.4 — UML-діаграма класів для бази даних

Клас «Database» є класом для створення глобального об’єкта бази даних. Цей об’єкт звертається до параметрів конфігурації системи для отримання URI-підключення [20] до компонента бази даних. Завдяки цим параметрам він отримує змогу створити клас «SessionMaker», який є класом-фабрикою для створення класів «Session». Останній відповідає за створення своєрідних сесій з базою даних, які надають можливість виконувати всі операції в рамках визначених транзакцій, зберігати значення великими масивами, замість створення запитів на кожний рядок окремо та зберігати (кешувати) певні значення в пам’яті, виділеній для роботи головного додатку, для зменшення кількості звернень до бази даних.

На рисунку 4.5 зображено діаграму класів для модуля графічного інтерфейсу. На ній показано, що графічний інтерфейс додатку (не враховуючи візуалізацію) складається з 3 головних вікон: початкового, вікна конфігурації та вікна роботи з даними.

Початкове вікно надає змогу відкривати інші вікна роботи з програмою і показує індикатор завантаження програми.

Вікно конфігурації надає можливість ввести необхідні параметри перед завантаженням програми, а також зберегти поточну конфігурацію або завантажити попередньо створену.

Останнє — вікно роботи з даними, в ньому створена логіка взаємодії з модулями математичної частини системи. На діаграмі видно залежність від інтерфейсів з центрального модуля. Такий підхід називається «Dependency inversion», його суть полягає в тому, що потрібно з’єднувати одні класи не напряму з іншими, а з інтерфейсами, які вони реалізують. Написання частин програми з таким підходом надає можливість легко замінювати частини коду при його розширенні в майбутньому.

На рисунку 4.6 можна побачити діаграму класів обгорток для роботи з базою даних і з файловою системою. Вони створені для уніфікації інтерфейсу роботи з даними. Таким чином, неважливо, звертається програма до бази чи до файлової системи — інтерфейс для взаємодії буде однаковий.

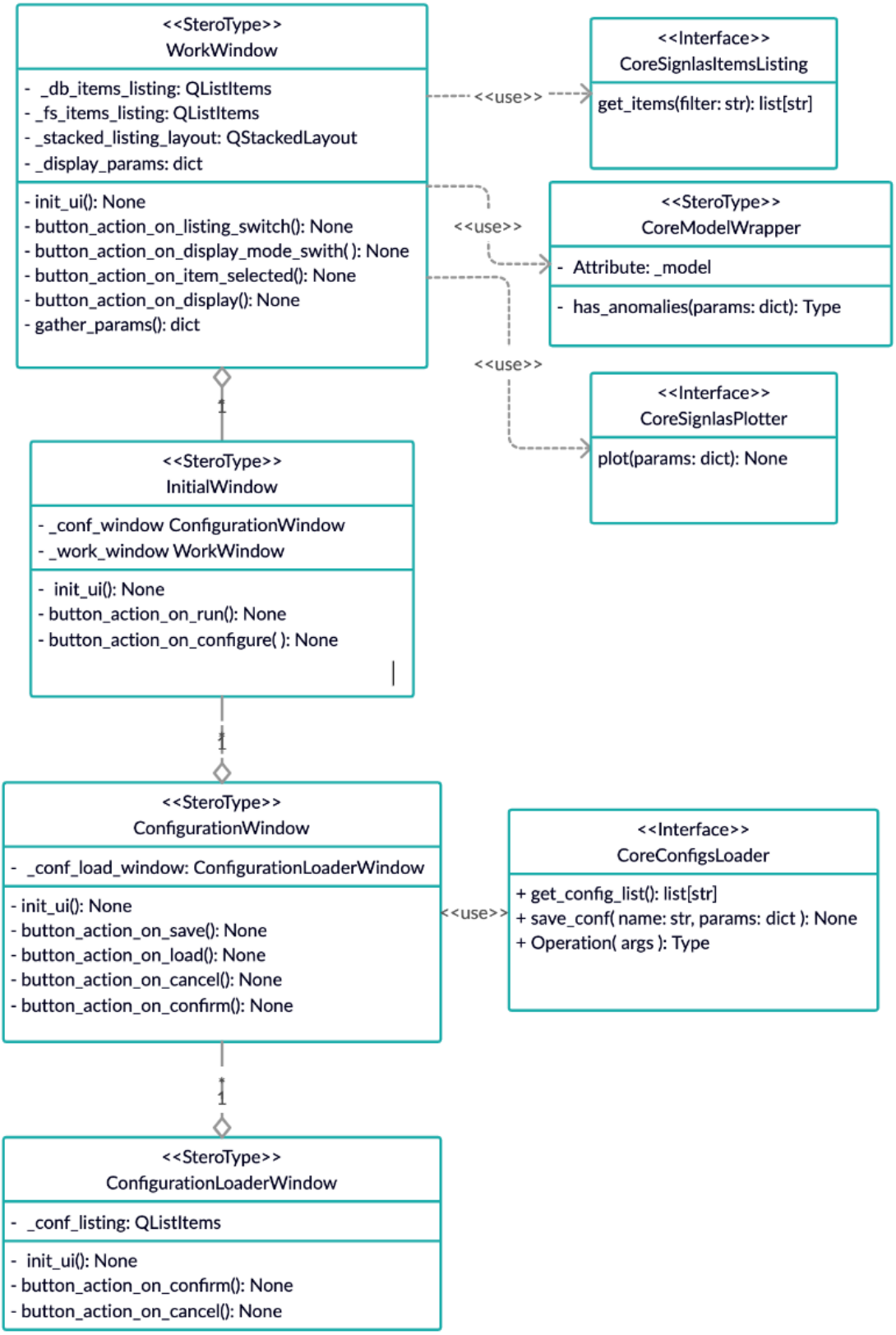


Рисунок 4.5 — UML-діаграма UI-компонентів

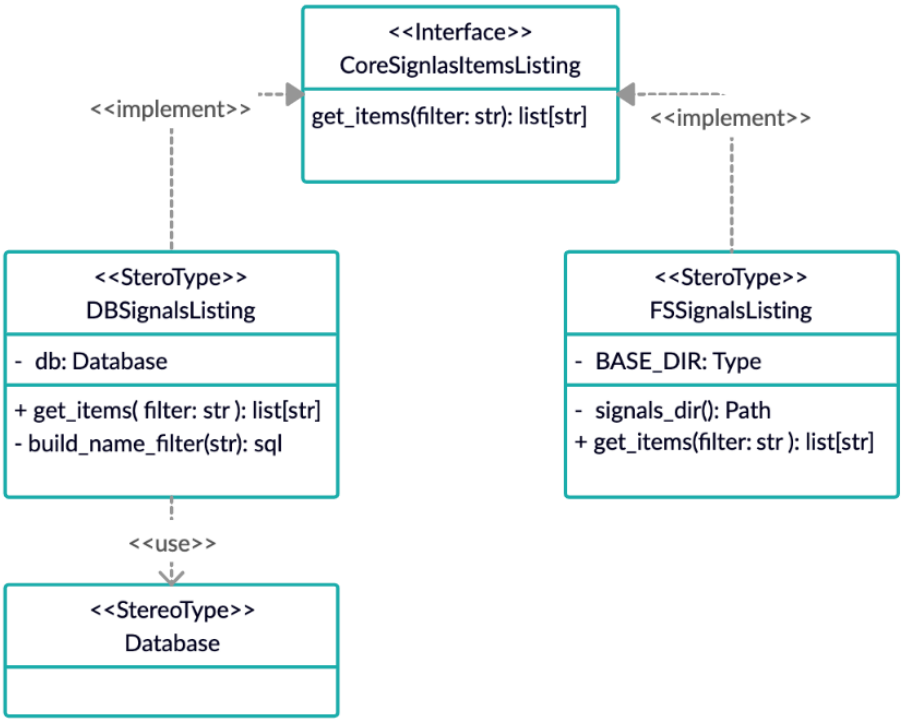


Рисунок 4.6 — UML-діаграма класів лістингу сигналів

Класи для роботи з конфігурацією та взаємодією з моделлю машинного навчання подано на рисунку 4.7.

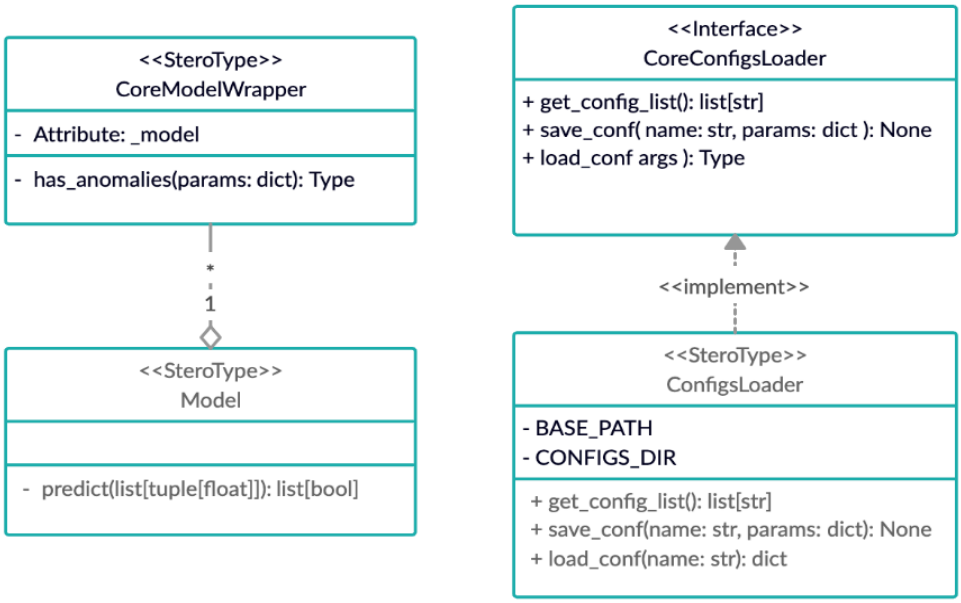


Рисунок 4.7 — UML-діаграма модулів конфігурації і моделі

Класи, зображені на діаграмах, реалізують попередньо вказані (рисунок 4.5) інтерфейси, щоб налаштувати мало залежну взаємодію між модулем графічного відображення та основною логікою програми, це надає можливість легко замінити модель машинного навчання або завантажувач конфігурації в даній системі.

Архітектуру програмної частини даної системи написано за принципами «SOLID», які є загальноприйнятою основою для написання коду в об’єктно-орієнтованому стилі. Дотримання даних принципів забезпечує можливість відносно нескладного розширення системи в майбутньому.

Із вище сказаного можна побачити, що програмний продукт написано відповідно до сучасних норм проектування та є легко розширюваним для забезпечення можливості подальшого розвитку продукту.

5. РОБОТА КОРИСТУВАЧА З ПРОГРАМНИМ ПРОДУКТОМ

Для роботи з розробленим програмним продуктом було створено три інтерфейси взаємодії та команди для розгортання проекту. Оскільки система побудована з використанням контейнерів і кількох процесів інтерпретатора, важливо враховувати можливості комп’ютера, який буде з нею взаємодіяти.

5.1 Системні вимоги

Для роботи з програмним засобом для обробки гідроакустичних сигналів необхідно:

— 32 Гб вільного місця на диску для розгортання системи;

— Unix-подібна операційна система;

— процесор Intel Core i3;

— 8 Гб оперативної пам’яті;

— додатково встановлене програмне забезпечення: docker, docker-compose, Python.

5.2 Розгортання програмного продукту

Для розгортання програмного продукту необхідно завантажити дані з гіт-репозиторію, після завантаження треба імпортувати необхідні змінні середовища, зайти в директорію і виконати такі команди:

— make init для ініціалізації;

— make gui-start для старту графічного інтерфейсу;

— make interactive-shell для взаємодії з інтерактивною Python-консоллю.

Команда для ініціалізації автоматично завантажить необхідні контейнери і запустить їх. Для перевірки статусу роботи контейнерів може бути використана команда docker-compose ps, для зупинки або старту роботи контейнерів також можна використати команди docker-compose start/stop/restart.

5.3 Результати роботи програмного продукту

Для роботи з продуктом використовуються три створені інтерфейси.

Графічний інтерфейс користувача є стандартним інтерфейсом і використовується для базової взаємодії з продуктом. Після запуску графічного інтерфейсу користувачу відображається початкове вікно програмного продукту (рисунок 5.1), на якому є варіанти подальшої взаємодії з продуктом, це — конфігурування, вихід та запуск основного робочого вікна.

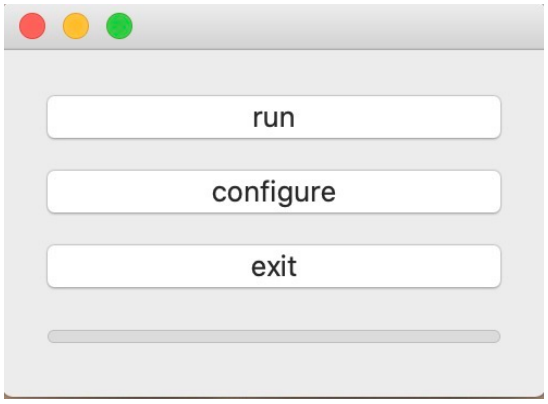


Рисунок 5.1 — Стартове вікно

Користувач може вибрати будь-який з запропонованих варіантів. За умови вибору варіанту “cconfigure” для користувача відкривається нове вікно конфігурації програмної системи. На якому є такі поля:

— HZ limit — верхній ліміт оброблюваних частот;

— Signals folder — папка, в якій розміщено файли з сигналами:

— Filename template— шаблон назви файла сигналу;

— Data prefix — префікс файла з сигналом;

— Prefix format — формат префікса файлів;

— Prefix length— довжина префікса файлів;

— Data format — формат, в якому записано дані;

— Sampling frequency — частота дискретизації;

— DB uri — рядок-адреса для підключення до бази даних.

Вікно конфігурування проекту подано на рисунку 5.2.

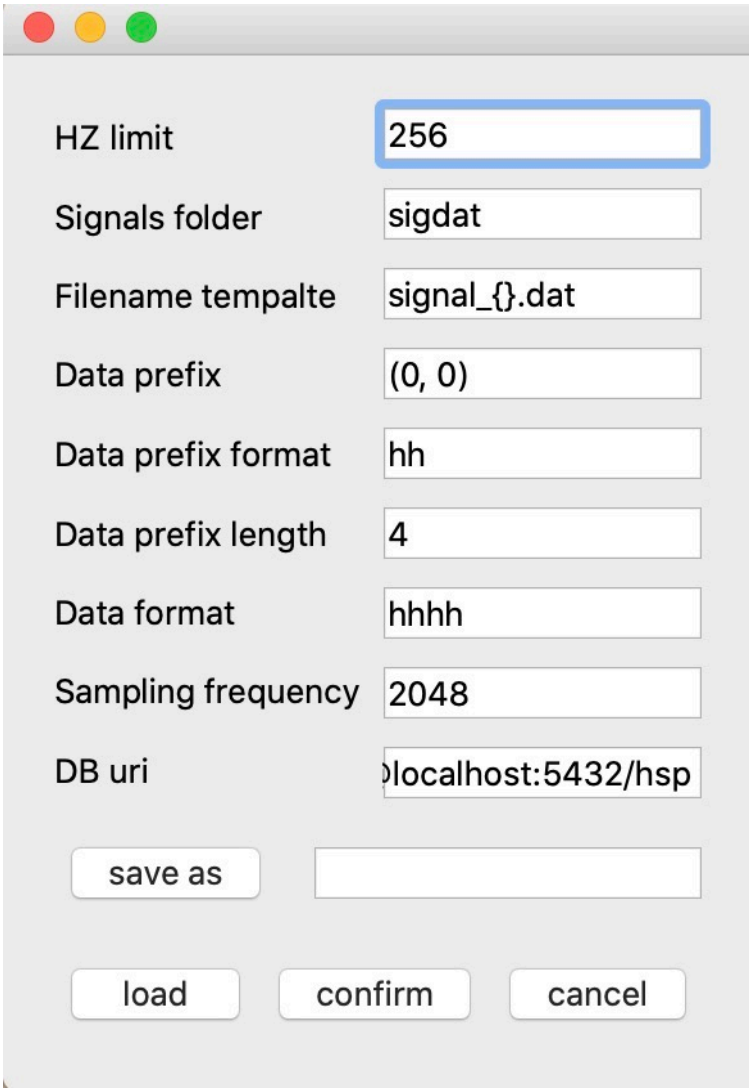


Рисунок 5.2 — Вікно конфігурації

Користувач також має можливість зберегти задану конфігурацію під вказаним у відповідному полі ім’ям. Важливо знати, що за умови вказання вже існуючого імені, нові значення конфігурації замінять старі у відповідному файлі. Також користувач має змогу завантажити вже збережену конфігурацію. Для завантаження користувачу потрібно натиснути клавішу “load”, після цього відкриється вікно з уже існуючими конфігураціями. Дане вікно подано на рисунку 5.3.

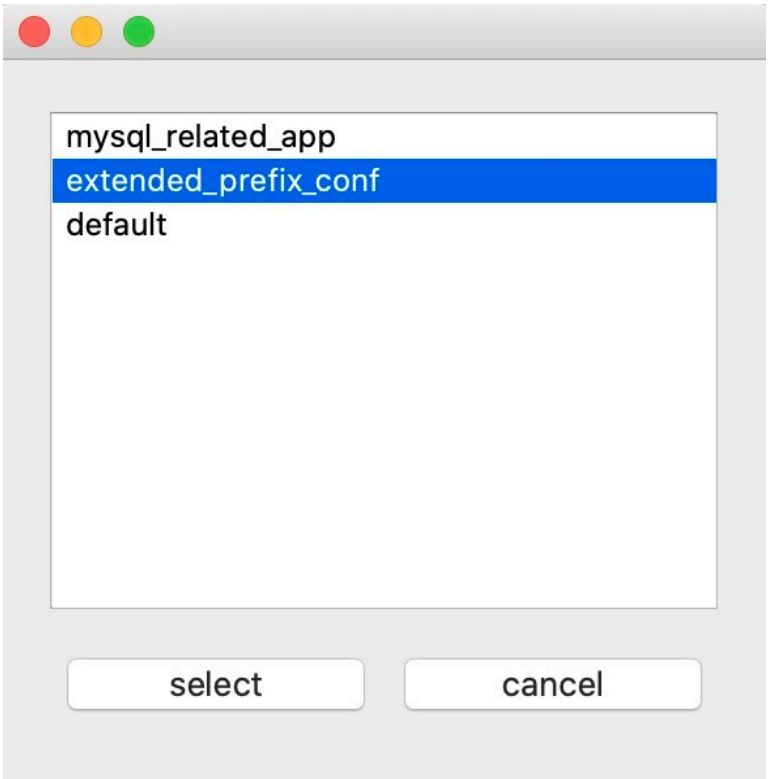


Рисунок 5.3 — Вікно завантаження конфігурації

Програма за замовчуванням завжди запускається з конфігурацією, записаною з іменем “default”. За необхідності, користувач може переписати параметри конфігурації за замовчуванням. Для цього їх треба ввести в вікні конфігурації і зберегти під іменем “default”.

Після конфігурування системи користувач може повернутися до стартового вікна і натиснути клавішу “run”, після чого почнуть завантажуватися необхідні для

програми модулі. За умови правильного розгортання продукту, завантаження модулів почнеться успішно і користувач побачить відповідну лінію завантаження в стартовому вікні.

Приклад успішного стару процесу завантаження подано на рисунку 5.4.

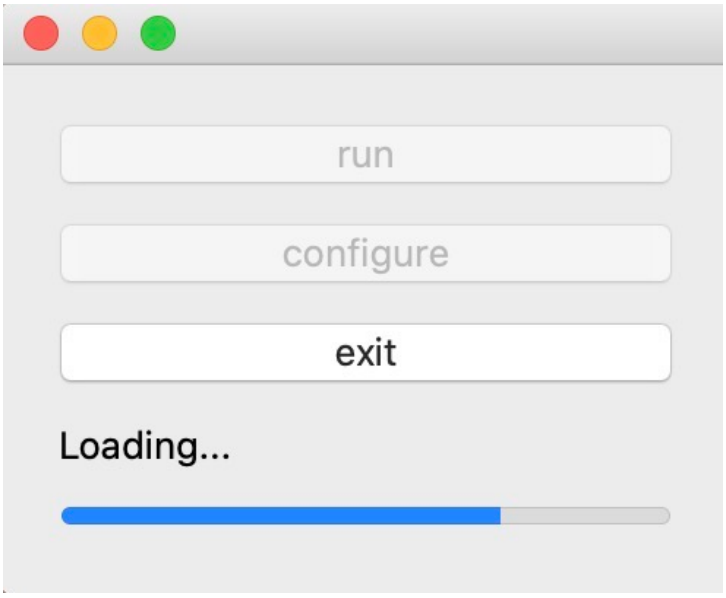


Рисунок 5.4 — Успішний процес завантаження модулів

Важливо пам’ятати, що всі клавіші, крім виходу, блокуються при завантаженні основного компонента програми. Це зроблено для безпеки і та запобігаггя процесу зміни конфігурації.

Після успішного завантаження користувач побачить основне вікно роботи з програмою (рисунок 5.5).

За замовчуванням вікно в першу чергу відображає файли сигналів, які розміщуються у вказаній в конфігурації директорії, але можна перемкнути програму в режим роботи з базою даних. Для цього користувачеві потрібно перемкнути в верхній частині вікна режим з “Folder” на “Database”. Після цього програма почне перевіряти наявність записів в базі даних.

На рисунку 5.6 подано вікно в режимі роботи з базою даних.

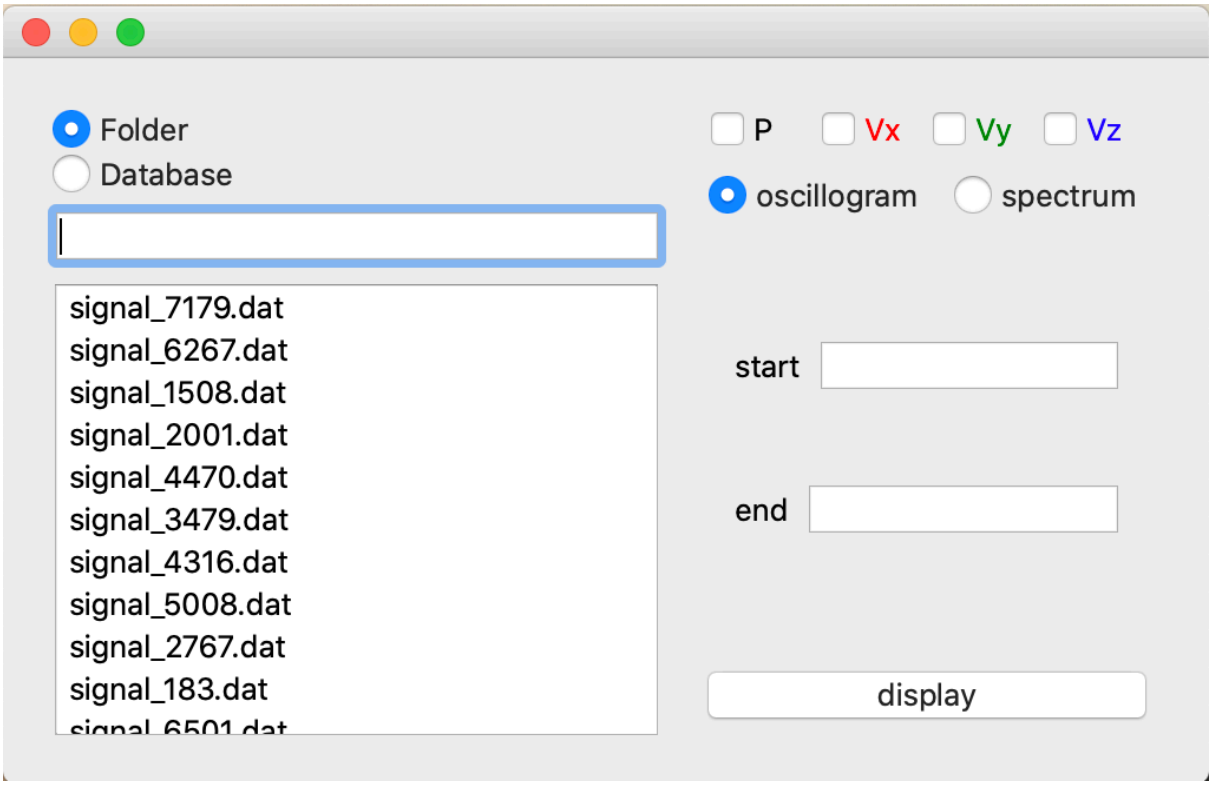


Рисунок 5.5 — Основне вікно роботи з програмою

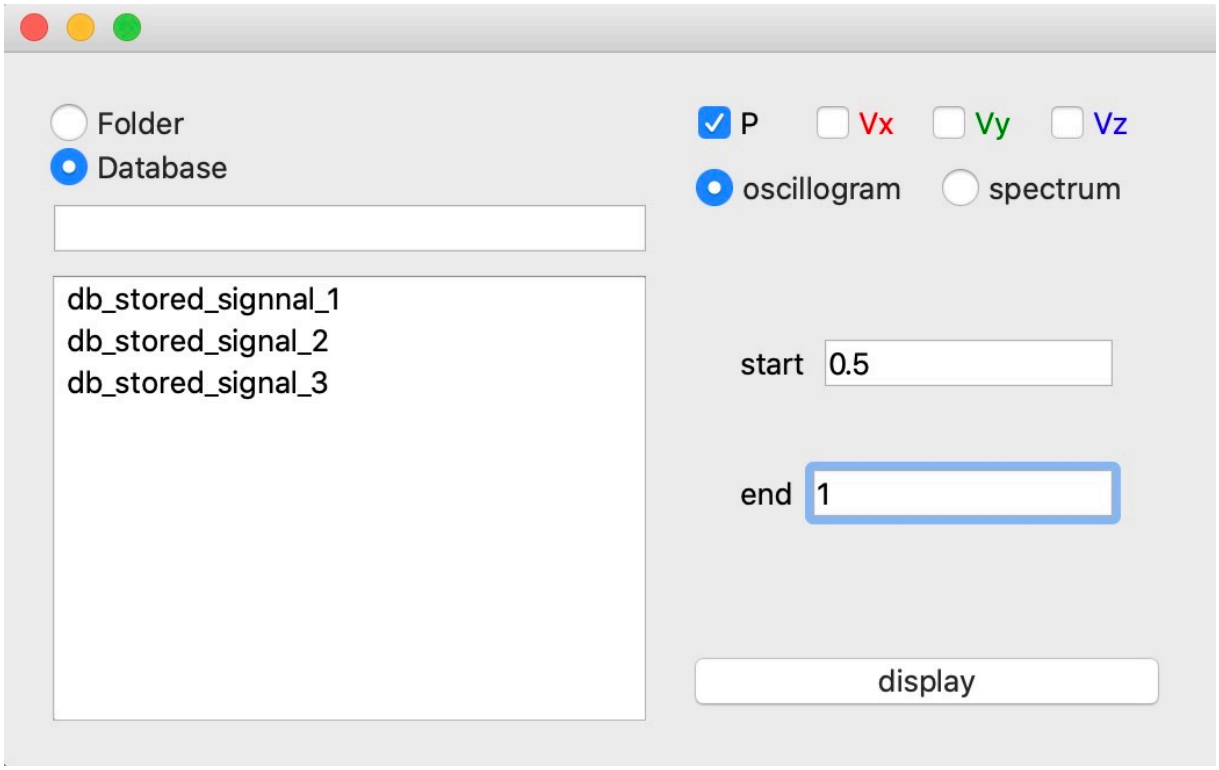


Рисунок 5.6 — Вікно в режимі роботи з базою даних

Оскільки записів в базі даних або файлів у вказаній директорії може бути дуже велика кількість, програма реалізує пошук за іменем у списку доступних сигналів. Приклад пошуку зображено на рисунку 5.7. Пошук працює однаково в будь-якому з режимів роботи.

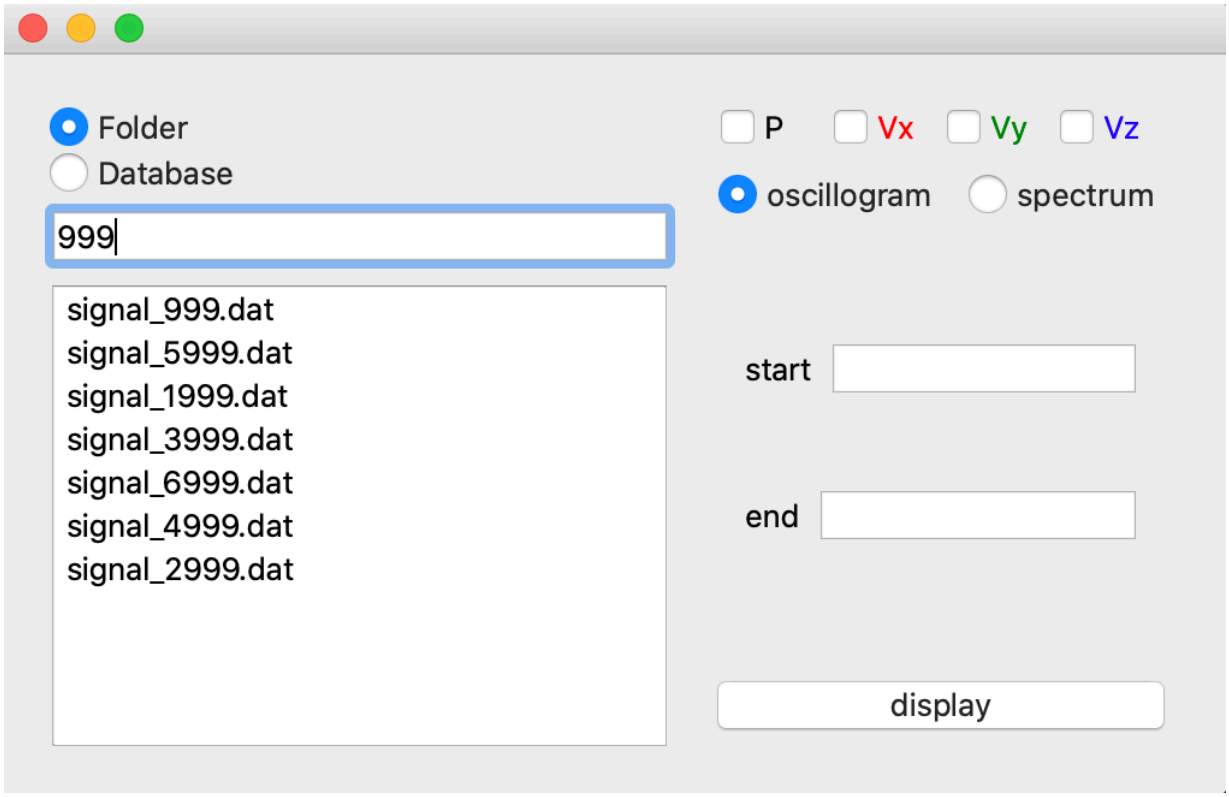


Рисунок 5.7 — Вікно з заданим пошуком за назвами

За замовчуванням вікно відкривається з вибором осцилограми. Після вибору запису сигналу з файла або з бази даних, користувачеві надається можливість його візуалізації. Користувач має можливість вказати початок і кінець бажаного для відображення проміжку. Також, користувач може вибрати, які саме зі значень він хоче побачити на графіку. Можливо вибрати до чотирьох значень включно, але не менше одного. Приклад відображення осцилограми по одному значенню на проміжку від нульової до другої секунди можна побачити на рисунку 5.8. На рисунку 5.9 подано приклад відображення сигналу відразу за трьома значеннями.

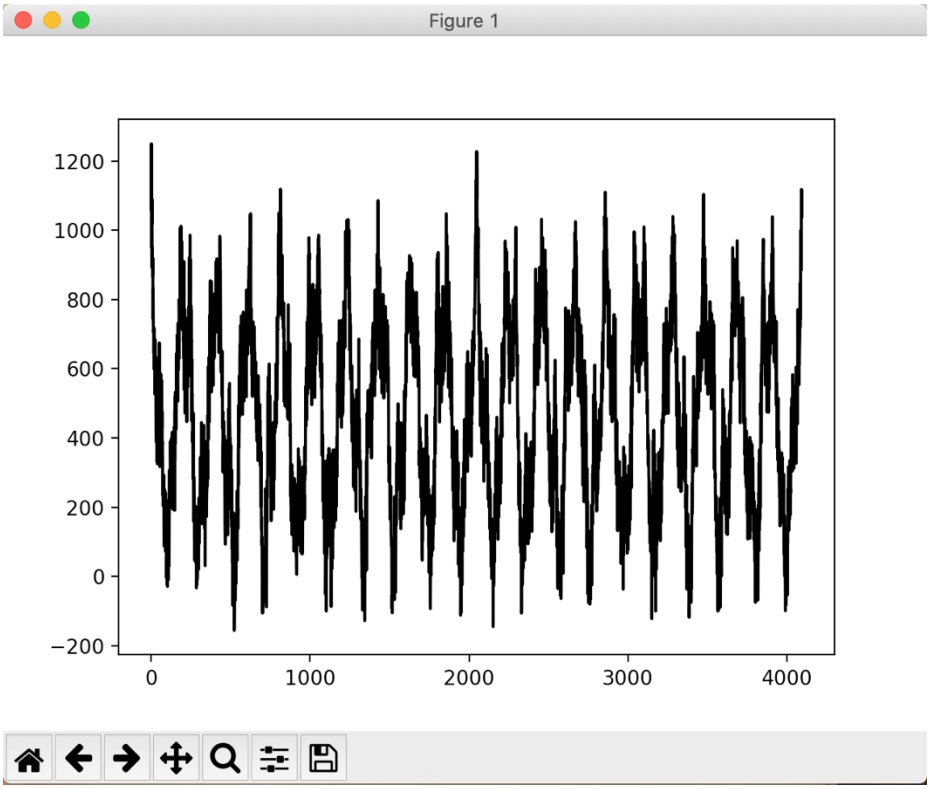


Рисунок 5.8 — Відображення осцилограми за одним значенням

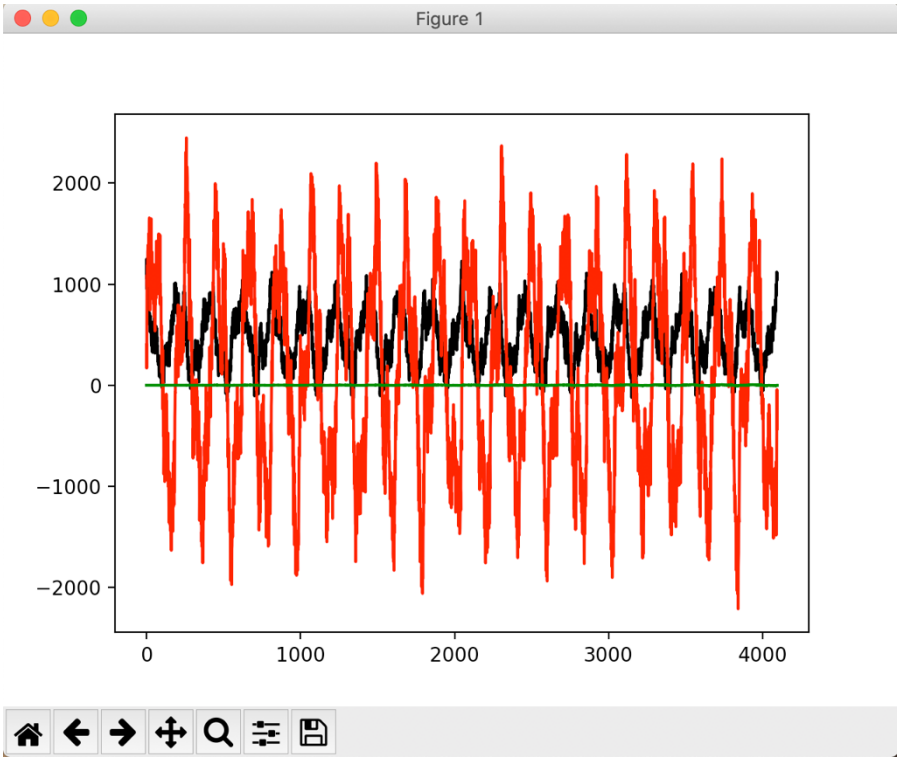


Рисунок 5.9 — Відображення осцилограми за трьома значеннями

Відображенням на даних графіках можна вільно маніпулювати, збільшувати і зменшувати масштаб, рухатися по графіку, зберігати, відновлювати відображення за замовчуванням. Приклад маніпуляції з зображенням, а саме зменшення масштабу, зображено на рисунку 5.10.

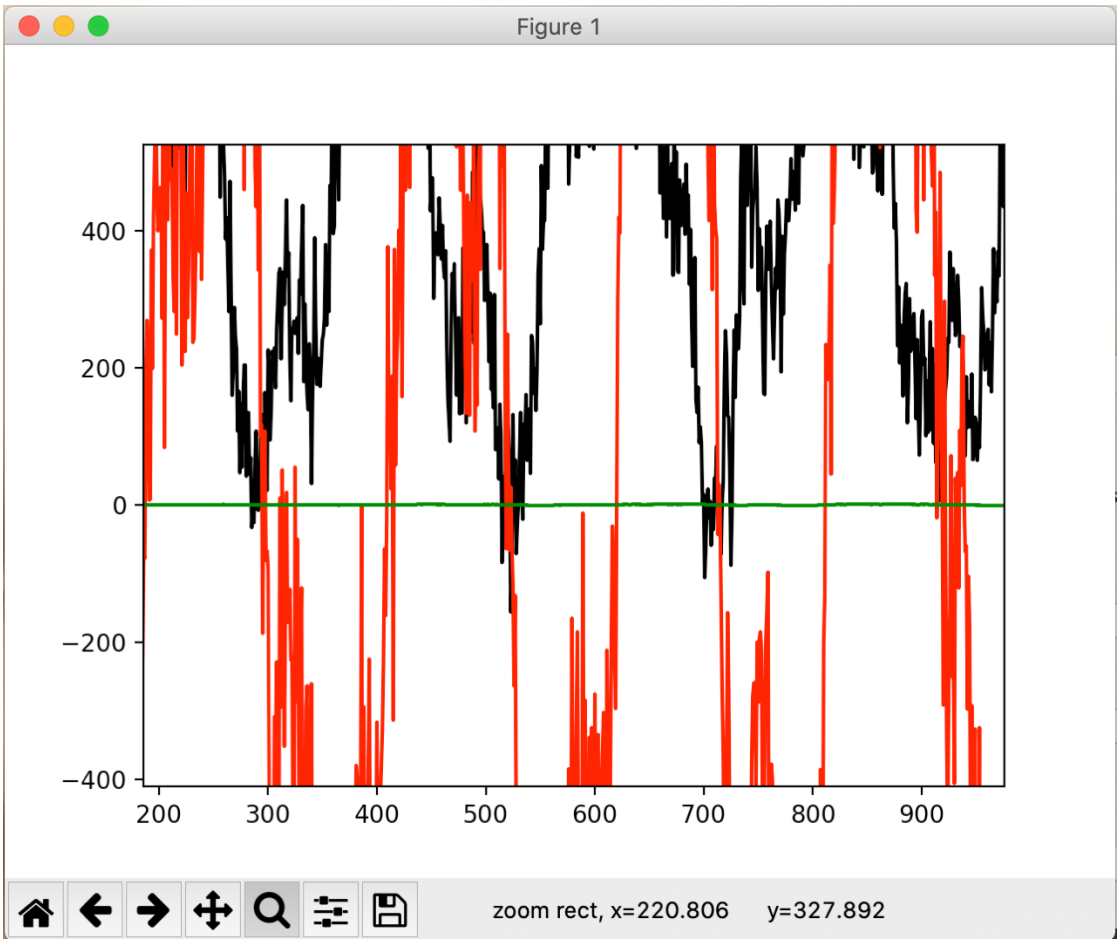


Рисунок 5.10 — Маніпуляція з зображенням

Користувач може перемкнутися в режим відображення спектру. Для спектру діють відповідні правила відображення. На рисунку 5.11 подано вікно з вказаними параметрами для відображення спектру.

Після вибору певного сигналу і параметрів відображення програма може відобразити спектр сигналу. Спектр сигналу для одного значення показаний на рисунку 5.12. Спектр сигналу за багатьма значеннями показаний на рисунку 5.13.

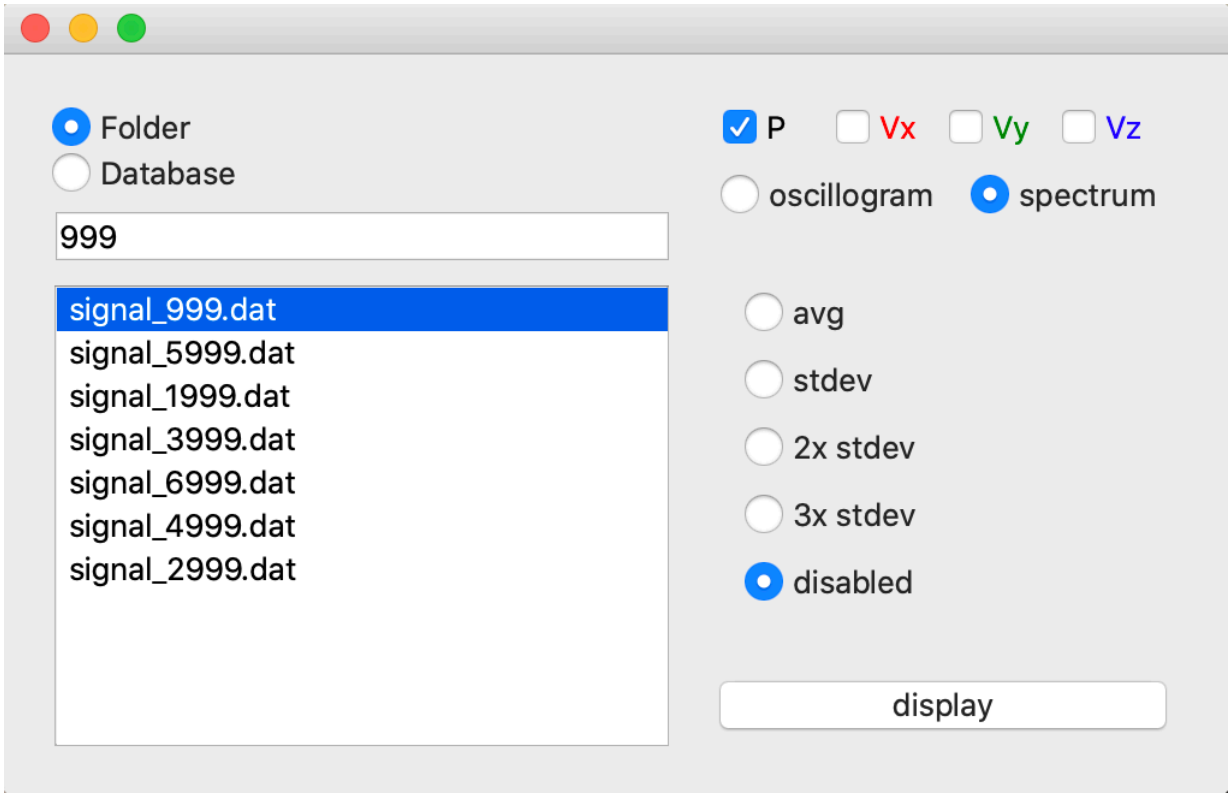


Рисунок 5.11 — Параметри для відображення спектру

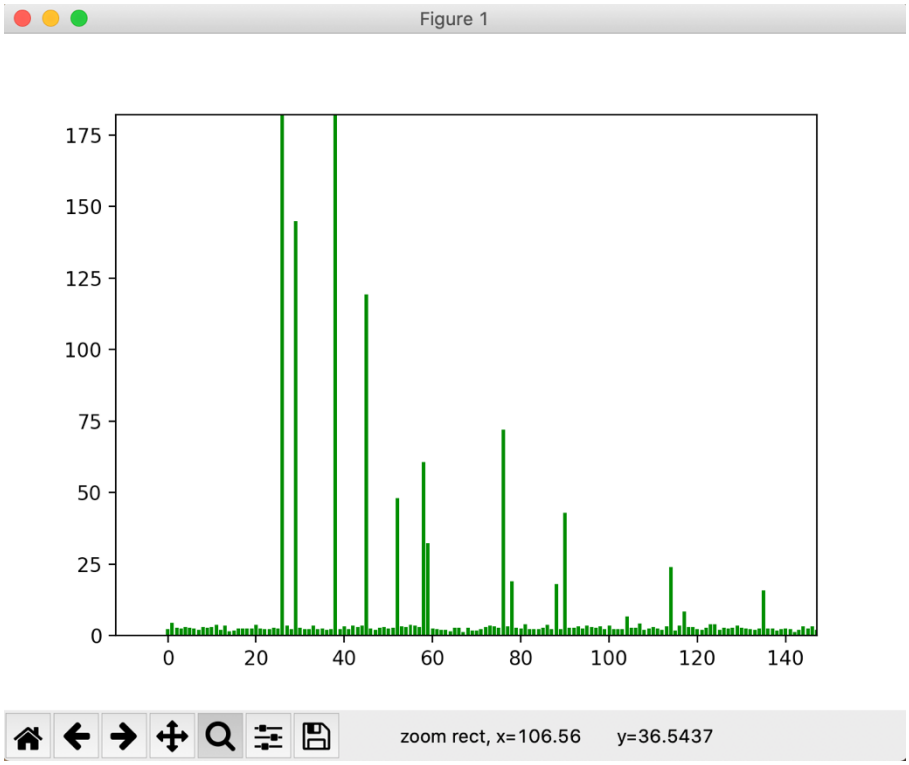


Рисунок 5.12 — Відображення спектру по одному значенню

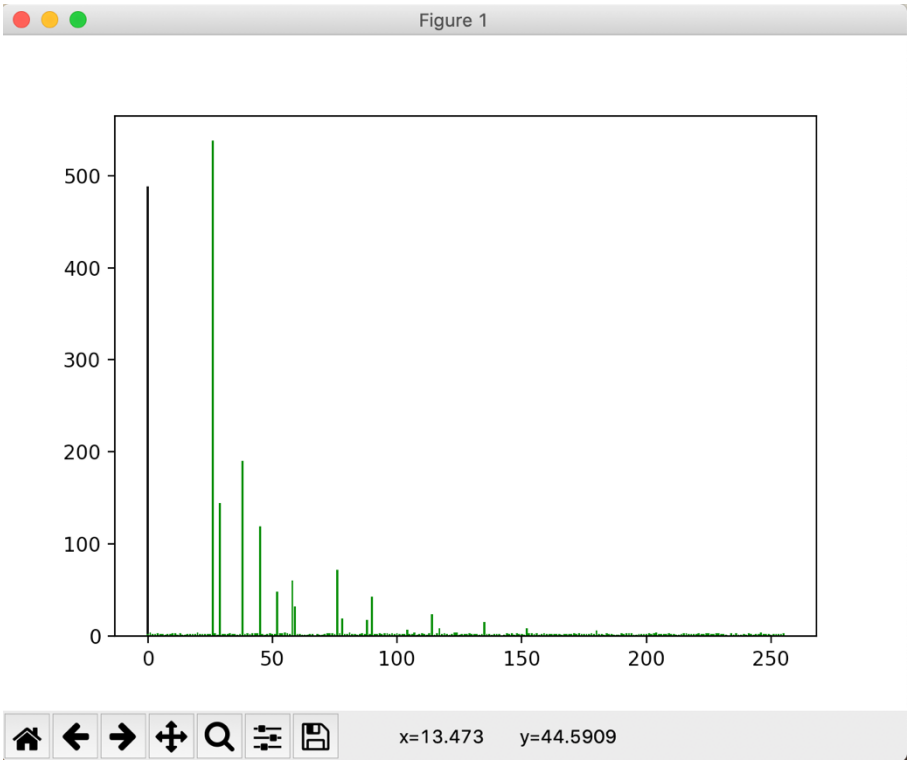


Рисунок 5.13 — Відображення спектру по багатьом значенням

З графіками спектрів можна виконувати всі ті самі маніпуляції, що і з осцилограмою. Варто зазначити, що запуск нового графіка не закриває старі, відображені до цього, тому користувач має можливість відображати й порівнювати багато графіків одночасно.

Крім графічного користувацького інтерфейсу, за замовчуванням користувач також має можливість скористатися Makefile-інтерфейсом (рисунок 5.14) та Jupyter-інтерфейсом (рисунок 5.15).

Варто зазначити, що дані інтерфейси є менш обмеженими й набагато більш гнучкими, ніж графічний користувацький інтерфейс. Вони надаються можливість використовувати розроблені в системі компоненти на функції напряму, пишучи операції відразу мовою Python, тобто при роботі з ними, користувач залишає за собою можливість виконувати будь-яку операцію, яка є синтаксично коректною для мови Python. За умови використання Jupyter-інтерфейсу, користувач отримує можливість запису всіх виконаних дій, що дає можливість зберегти певний прогрес роботи, закрити

програмний продукт і після паузи ввімкнути його знову, продовживши роботу з місця, на якому її було зупинено.

```
Python 3.8.3 (v3.8.3:6f8c8320e9, May 13 2020, 16:29:34)
Type 'copyright', 'credits' or 'license' for more information
IPython 7.15.0 -- An enhanced Interactive Python. Type '?' for help.

In [1]: from data_utils import load_by_id

In [2]: ds = load_by_id(742)

In [3]: from spectrum import spectrum

In [4]: s_ds = spectrum(ds.p)

In [5]: s_ds
Out[5]:
array([4.72435135e+02, 1.76273880e-01, 1.40879807e-01, 8.24503743e-01,
       6.25918094e-01, 4.33152238e-01, 2.88242664e-01, 2.10084691e-01,
       6.77498449e-01, 3.55038888e-01, 1.23506907e-01, 4.52786337e-01,
       5.00840656e-01, 5.07944001e-01, 3.45901635e-01, 7.43742178e-01,
       2.82286193e-01, 2.41048367e-01, 3.88045472e-01, 4.77619394e-01,
```

Рисунок 5.14 — Makefile-інтерфейс

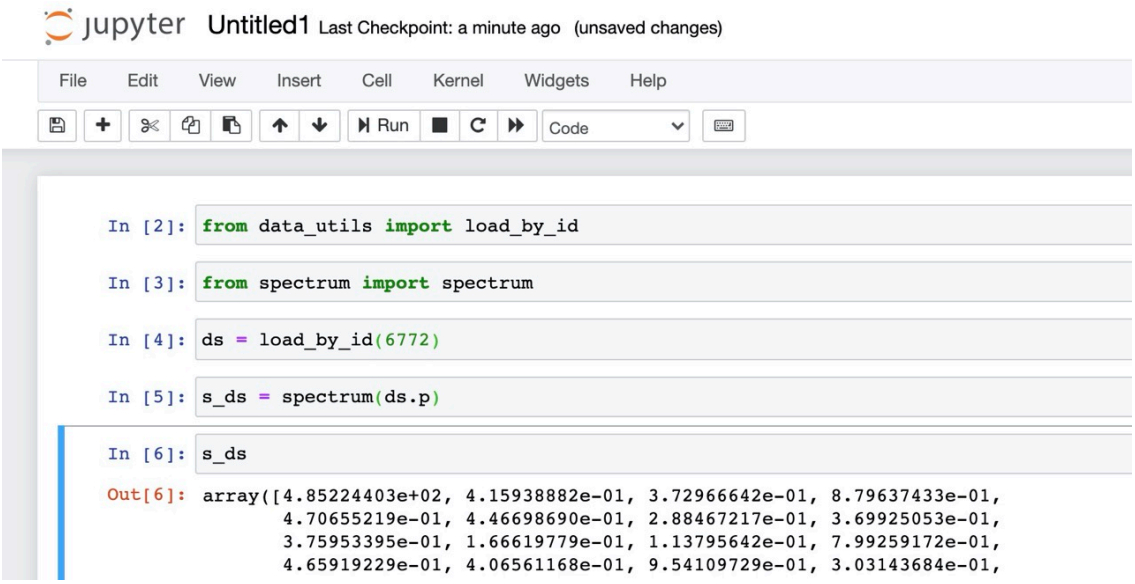


Рисунок 5.15 — Jupyter-інтерфейс

Завдяки такій гнучкості системи користувач може відображати необхідні йому графіки в інших нотаціях. Так, наприклад, користувач має можливість відобразити в

одному вікні спектр сигналу відразу за всіма значеннями на різних координатних осях. Також користувач може замінити, вказане за замовчуванням, відображення спектру в вигляді гістограми на неперервну функцію. На рисунку 5.18 показано відображення спектру сигналу в інших нотаціях.

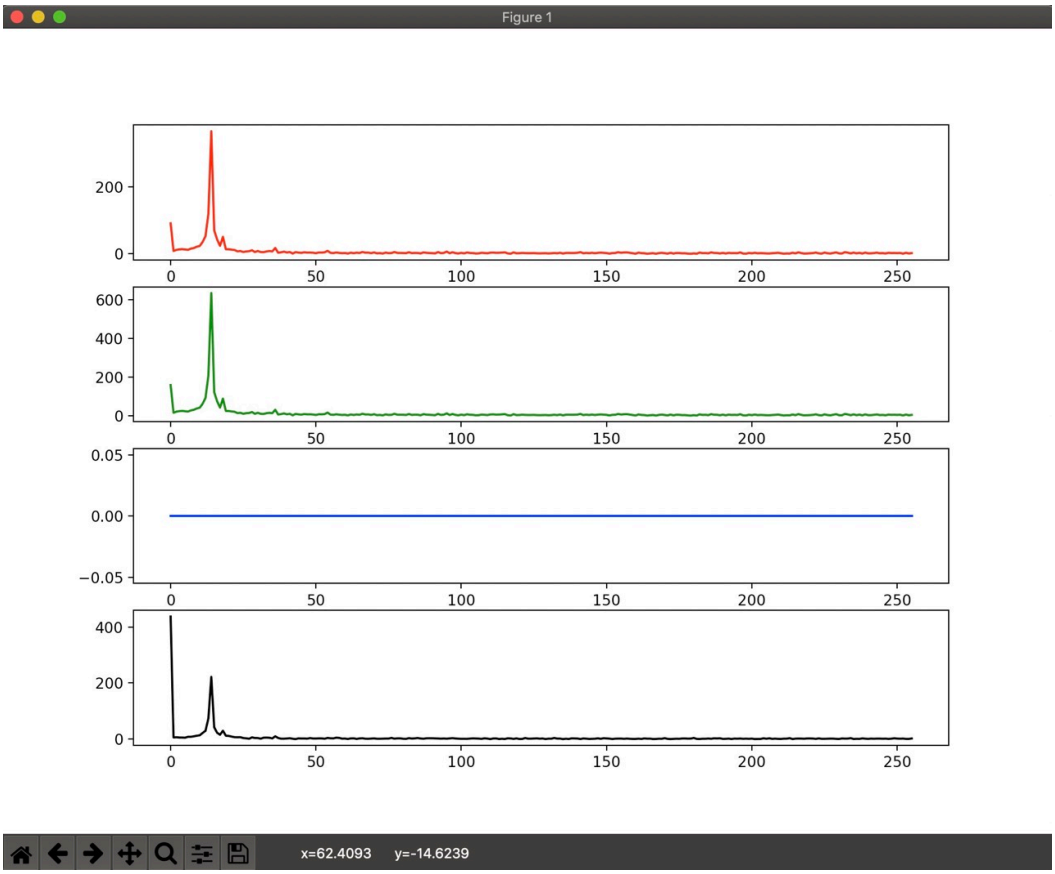


Рисунок 5.18 — Спектр сигналу

Три створені інтерфейси надають можливість гнучкого користування програмою незалежно від первинної потреби. Кожен з них призначається для специфічних випадків взаємодії. Так, головний графічний інтерфейс — для виконання базових операцій щодо обробки, візуалізації й аналізу сигналу. Інтерфейси Makefile і Jupyter — для взаємодії з програмними функціями напряму завдяки вбудованому інтерпретатору Python.

ВИСНОВКИ

При виконанні дипломної роботи розроблено програмний продукт, призначений для інтелектуально-статистичної обробки гідроакустичних сигналів. Створено інтерфейси взаємодії з програмним продуктом, до яких належать графічний користувацький інтерфейс з використанням пакету MatPlotLib, Makefile-інтерфейс і Jupyter-інтерфейс. Створено й інтегровано до проекту модель машинного навчання для виявлення аномалій на основі методу класифікації “Decision Tree”. Додана можливість візуалізації даних у побічних процесах мови Python за допомогою бібліотеки MatPlotLib.

Створена програмна система є гнучкою для розширення і може бути використана як основа для більш точних вузьконаправлених рішень.

Користувачами системи можуть бути як студенти навчальних закладів, використовуючи її для поглиблення знань про гідроакустику, так і працівники сфери гідроакустики для проведення аналізу та обробки сигналів.

Результати роботи доповідалися на VII Науково-практичній дистанційній конференції фахівців з розробки програмного забезпечення (VII Scientific and Practical Virtual Conference of Software Development Specialists) 2020 року [21].

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Румынская И.А. Основы гидроакустики / И.А. Румынская. — Л.: Судостроение, 1979. — 209 с.

2. Годиенко В.А. Векторно-фазовые методы в акустике / В.А. Годиенко.— М.: Физматлит, 2007. — 480 с.

3. Muller A.C. Introduction to Machine Learning with Python: A Guide for Data Scientists / A.C. Muller, S. Guido. — 2016. — 392 p.

4. Duran B.S. Cluster Analysis. A Survey / B.S. Duran, P.L. Odell. — New York: Springer-Verlag Berlin Heidelberg, 1974. — 140 p.

5. Fluch P. Machine learning. The Art and Science of Algorithms that Make Sense of Data / Peter Fluch. — Cambridge: Cambridge University Press, 2012. — 396 p.

6. Git Book [Електронний ресурс]. — 2014. — Режим доступу до ресурсу: https://git-scm.com/book/uk/v2.

7. Mouat A. Using Docker / Adrian Mouat. — Boston: O'Reilly Media, Inc., 2016. — 337 p.

8. Моргунов Е.П. — PostgreSQL. Основы языка SQL: учеб. пособие / Е.П. Моргунов; под ред. Е.В. Рогова, П.В. Лузанова. — СПб.: БХВ-Петербург, 2018. — 336 с.

9. Raschka S. Python Machine Learning Second Edition / S. Raschka, V. Mirjalili. — Birmingham: Packt Publishin, 2017. — 840 p.

10. Шпак З.Я. Програмування мовою С / З.Я. Шпак. — Львів: Оріяна-Нова, 2006. — 431 с.

11. Index TIOBE [Електронний ресурс]. — 2020. — Режим доступу до ресурсу: https://www.tiobe.com/tiobe-index/.

12. Groff J.R. SQL. The Complete Reference / J.R. Groff, P.N. Weinberg, A.J. Oppel. — New York: The McGraw-Hill, 2010. — 882 p.

13. Summerfield M. Advanced Qt Programming / Mark Summerfield. — Boston: Addison-Wesley, 2010. — 537 p.

14. Galitz W.O. The Essential Guide to User Interface Design: An Introduction to GUI Design Principles andTechniques / Wilbert Galitz. — Indianapolis: Wiley Publishing, Inc, 2007. — 857 p.

15. Phillips D. Python 3 Object Oriented Programming / Dusty Phillips. — Birmingham: Packt Publishing, 2010. — 391 p.

16. Myers G.J. The Art of Software Testing / Glenford Myers. — Hoboken: John Wiley & Sons, Inc, 2004. — 151 p.

17. Martin R.C. Clean Architecture: A Craftsman’s Guide to Software Structure and Design / Robert Martin. — Boston: Pearson Education, Inc, 2018. — 421 p.

18. pathlib [Електронний ресурс]. — 2014. — Режим доступу до ресурсу: https://docs.python.org/3/library/pathlib.html.

19. pickle [Електронний ресурс] — Режим доступу до ресурсу: https://docs.python.org/3/library/pickle.html.

20. SQLAlchemy [Електронний ресурс]. — 2006. — Режим доступу до ресурсу: https://www.sqlalchemy.org/.

21. Mykhal’ko V.H. Software tools for intellectual and statistical processing of hydroacoustic signals / V.H. Mykhal’ko, L.I. Kublii // Modern Aspects of Software Development: Proceedings of VII Scientific and Practical Virtual Conference of Software Development Specialists, June, 1, 2020. – Kyiv: Igor Sikorsky KPI, 2020. — P. 31-35.

ДОДАТОК 1

Програмні засоби інтелектуально-статистичної обробки гідроакустичних сигналів

Специфікація

УКР.НТУУ”КПІ ім. Ігоря Сікорського”_ТЕФ_АПЕПС_ТМ62202_20Б

Аркушів 2

Київ – 2020

Позначення	Найменування	Примітки
Документація		
УКР.НТУУ«КПІ ім. Ігоря Сікорського»_ТЕФ_АПЕПС_ТМ 62202_20Б	Записка.docx	Пояснювальна записка
Компоненти		
УКР.НТУУ«КПІ ім. Ігоря Сікорського»_ТЕФ_АПЕПС_ТМ 62202_20Б 12-1	src/application_configuration/__init__.py	Модуль конфігурації програми
УКР.НТУУ«КПІ ім. Ігоря Сікорського»_ТЕФ_АПЕПС_ТМ 62202_20Б 12-2	src/core	Модулі центрального компоненту
УКР.НТУУ«КПІ ім. Ігоря Сікорського»_ТЕФ_АПЕПС_ТМ 62202_20Б 12-3	src/ui/work_window.py	Модуль головного вікна графічного інтерфейсу
УКР.НТУУ”КПІ ім. Ігоря Сікорського”_ТЕФ_АПЕПС_ТМ 62202_20 13-1	Опис.docx	Опис програми

ДОДАТОК 2

Програмні засоби інтелектуально-статистичної обробки гідроакустичних сигналів

Текст програми

УКР.НТУУ”КПІ ім. Ігоря Сікорського”_ТЕФ_АПЕПС_ТМ61178_20Б 12

Аркушів 9

Київ – 2020

```
import json
from typing import List
from constants import BASE_PATH
from decorators import singleton


CONF_PATH = BASE_PATH / ".conf"


@singleton
class Configuration:
    HZ_LIMIT: int = None
    SIGNALS_DIR: str = None
    DATA_FILENAME_TEMPLATE: str = None
    DATA_PREFIX: str = None
    DATA_PREFIX_FORMAT: str = None
    DATA_PREFIX_LENGTH: int = None
    DATA_FORMAT: str = None
    SAMPLING_FREQUENCY: int = None
    DB_URI: str = None
    DEFAULT_MODEL_PATH = BASE_PATH / "model.pickle"

    def set_values(self, values: dict):
        for key, value in values.items():
            setattr(self, key, value)
        if isinstance(self.DATA_PREFIX, str):
            self.DATA_PREFIX = eval(self.DATA_PREFIX)
        self.HZ_LIMIT = int(self.HZ_LIMIT)
        self.DATA_PREFIX_LENGTH = int(self.DATA_PREFIX_LENGTH)
        self.SAMPLING_FREQUENCY = int(self.SAMPLING_FREQUENCY)

    @property
    def dict(self):
        return {
            "HZ_LIMIT": int(self.HZ_LIMIT),
```

```
            "SIGNALS_DIR": self.SIGNALS_DIR,
            "DATA_FILENAME_TEMPLATE": self.DATA_FILENAME_TEMPLATE,
            "DATA_PREFIX": eval(self.DATA_PREFIX) if isinstance(self.DATA_PREFIX, str) else self.DATA_PREFIX,
            "DATA_PREFIX_FORMAT": self.DATA_PREFIX_FORMAT,
            "DATA_PREFIX_LENGTH": int(self.DATA_PREFIX_LENGTH),
            "DATA_FORMAT": self.DATA_FORMAT,
            "SAMPLING_FREQUENCY": int(self.SAMPLING_FREQUENCY),
            "DB_URI": self.DB_URI,
        }


def load_configuration(conf_name: str) -> Configuration:
    path = CONF_PATH / f"{conf_name}.json"
    with open(path) as f:
        configuration.set_values(json.loads(f.read()))
    return configuration



def save_configuration(conf_name: str) -> None:
    path = CONF_PATH / f"{conf_name}.json"
    with open(path, "w") as f:
        f.write(json.dumps(configuration.dict, indent=4))



def available_configurations() -> List[str]:
    return [conf.stem for conf in CONF_PATH.iterdir() if conf.suffix == ".json"]



configuration = Configuration()
load_configuration("default")
```



```
def plot(kwargs):
    kwargs["conf"] = configuration.dict
    process = Process(target=_plot, kwargs=kwargs)
    process.start()
```

```
def _plot(**kwargs):
    import numpy as np
    import pandas as pd
    from core import volume_data_utils
    from application_configuration import configuration
    from matplotlib import pyplot
    from core.spectrum import sampling
    configuration.set_values(kwargs["conf"])
    if kwargs["location"].lower() == "folder":
        df = volume_data_utils.load_by_name(kwargs["label"])
    else:
        df = pd.DataFrame()

    if kwargs["graph_type"] == "oscillogram":
        start = float(kwargs["start"] or 0)
        end = float(kwargs["end"] or 10)
        if not (0 <= start <= 10):
            start = 0
        if not (0 <= end <= 10):
            end = 10
        if start >= end:
            start, end = 0, 10
        index_start = int(start * len(df) / 10)
        index_end = int(end * len(df) / 10)
        df = df.iloc[index_start:index_end]
        if kwargs["p"]:
            pyplot.plot(range(index_start, index_end), df.p, color="k")
        if kwargs["vx"]:
            pyplot.plot(range(index_start, index_end), df.vx, color="r")
        if kwargs["vy"]:
            pyplot.plot(range(index_start, index_end), df.vy, color="g")
        if kwargs["vz"]:
            pyplot.plot(range(index_start, index_end), df.vz, color="b")
        pyplot.xlabel("t, 1/2048s")
        pyplot.ylabel("Y(t)")
        pyplot.show()
    else:
```

```
py, vxy, vyy, vzy = None, None, None, None
colors = ("k", "r", "g", "b")
if kwargs["p"]:
    py, px, plt = pyplot.magnitude_spectrum(df.p, Fs=configuration.SAMPLING_FREQUENCY)
    py = list(sampling(np.column_stack((px, py)), configuration.HZ_LIMIT))
if kwargs["vx"]:
    vxy, vxx, plt = pyplot.magnitude_spectrum(df.vx, Fs=configuration.SAMPLING_FREQUENCY)
    vxy = list(sampling(np.column_stack((vxx, vxy)), configuration.HZ_LIMIT))
if kwargs["vy"]:
    vyy, vyx, plt = pyplot.magnitude_spectrum(df.vy, Fs=configuration.SAMPLING_FREQUENCY)
    vyy = list(sampling(np.column_stack((vyx, vyy)), configuration.HZ_LIMIT))
if kwargs["vz"]:
    vzy, vzx, plt = pyplot.magnitude_spectrum(df.vz, Fs=configuration.SAMPLING_FREQUENCY)
    vzy = list(sampling(np.column_stack((vzx, vzy)), configuration.HZ_LIMIT))
pyplot.close()
for y, color in zip((py, vxy, vyy, vzy), colors):
    if y is not None:
        pyplot.bar(range(configuration.HZ_LIMIT), list(y), color=color)
delimiter_resolver = {
    "avg": np.average,
    "stdev": lambda y: np.average(y) + np.std(y),
    "2x stdev": lambda y: np.average(y) + 2*np.std(y),
    "3x stdev": lambda y: np.average(y) + 3*np.std(y)
}
if kwargs["spectrum_delimiter"] in ("avg", "stdev", "2x stdev", "3x stdev"):
    for y, color in zip((py, vxy, vyy, vzy), colors):
        if y is not None:
            delim_value = delimiter_resolver[kwargs["spectrum_delimiter"]](y)
            y = np.empty(len(y))
            y.fill(delim_value)
            pyplot.plot(range(len(y)), y, c=color)
pyplot.xlabel("Hz")
pyplot.ylabel("Y(Hz)")
pyplot.show()
```

```
class SignalsListing(QListWidget):
    _listing: AbstractListing = None

    def __init__(self, *args, **kwargs):
        self._listing = kwargs.pop("listing")
        super().__init__(*args, **kwargs)
        self.update_listing("")

    def update_listing(self, filter_by: str):
        self.clear()
        self.addItems(self._listing.list_items(filter_by))
class WorkWindow(QWidget):
    _listing_folder: SignalsListing = None
    _listing_db: SignalsListing = None
    _input_search: QLineEdit = None

    _listing_stacked_layout: QStackedLayout = None

class WorkWindow(QWidget):
    _listing_folder: SignalsListing = None
    _listing_db: SignalsListing = None
    _input_search: QLineEdit = None

    _listing_stacked_layout: QStackedLayout = None

class WorkWindow(QWidget):
    _listing_folder: SignalsListing = None
    _listing_db: SignalsListing = None
    _input_search: QLineEdit = None




    _listing_stacked_layout: QStackedLayout = None
```

```
_radio_button_folder: QRadioButton = None
_radio_button_db: QRadioButton = None
_radio_button_group: QButtonGroup = None

_radio_button_oscillogram: QRadioButton = None
_radio_button_spectrum: QRadioButton = None
_radio_button_graph_type: QButtonGroup = None

_checkbox_dimension_p: QCheckBox = None
_checkbox_dimension_vx: QCheckBox = None
_checkbox_dimension_vy: QCheckBox = None
_checkbox_dimension_vz: QCheckBox = None
_graph_type_stacked_layout: QStackedLayout = None
_oscillogram_control_widget: QWidget = None
_spectrum_control_widget: QWidget = None

_radio_spectrum_filter_limit_avg: QRadioButton = None
_radio_spectrum_filter_limit_stdev: QRadioButton = None
_radio_spectrum_filter_limit_2stdev: QRadioButton = None
_radio_spectrum_filter_limit_3stdev: QRadioButton = None
_radio_spectrum_filter_limit_disabled: QRadioButton = None
_radio_spectrum_filter_group_limit: QButtonGroup = None

_button_draw: QPushButton = None

def __init__(self, *args, **kwargs):
    super().__init__(*args, **kwargs)
    self.init_components()
    self.setWindowTitle("Hydroacoustic signals analysis")

def init_components(self):

    main_layout = QHBoxLayout()

    listings_panel_layout = QVBoxLayout()
    self._listings_stacked_layout = QStackedLayout()
```

```
self._input_search = QLineEdit()
self._radio_button_folder = QRadioButton("Folder")
self._radio_button_folder.setChecked(True)
self._radio_button_db = QRadioButton("Database")
self._radio_button_group = QButtonGroup()
self._radio_button_group.addButton(self._radio_button_folder)
self._radio_button_group.addButton(self._radio_button_db)
self._radio_button_group.buttonClicked.connect(self.change_listing)
self._listing_folder = SignalsListing(listing=VolumeListing())
self._listing_db = SignalsListing(listing=DatabaseListing())
self._listings_stacked_layout.addWidget(self._listing_folder)
self._listings_stacked_layout.addWidget(self._listing_db)
self._listings_stacked_layout.setCurrentIndex(0)
listings_panel_layout.addWidget(self._radio_button_folder)
listings_panel_layout.addWidget(self._radio_button_db)
listings_panel_layout.addWidget(self._input_search)
listings_panel_layout.addLayout(self._listings_stacked_layout)

main_layout.addLayout(listings_panel_layout)

self._input_search.textChanged.connect(self.update_listings)

draw_parameters_layout = QVBoxLayout()

dimensions_layout = QHBoxLayout()
self._checkbox_dimension_p = QCheckBox("P")
self._checkbox_dimension_p.setStyleSheet("color: black")
self._checkbox_dimension_vx = QCheckBox("Vx")
self._checkbox_dimension_vx.setStyleSheet("color: red")
self._checkbox_dimension_vy = QCheckBox("Vy")
self._checkbox_dimension_vy.setStyleSheet("color: green")
self._checkbox_dimension_vz = QCheckBox("Vz")
self._checkbox_dimension_vz.setStyleSheet("color: blue")
dimensions_layout.addWidget(self._checkbox_dimension_p)
dimensions_layout.addWidget(self._checkbox_dimension_vx)
dimensions_layout.addWidget(self._checkbox_dimension_vy)
```

```
dimensions_layout.addWidget(self._checkbox_dimension_vz)
draw_parameters_layout.addLayout(dimensions_layout)

graph_type_layout = QHBoxLayout()
self._radio_button_oscillogram = QRadioButton("oscillogram")
self._radio_button_oscillogram.setChecked(True)
self._radio_button_spectrum = QRadioButton("spectrum")
self._radio_button_graph_type = QButtonGroup()
self._radio_button_graph_type.buttonClicked.connect(self.change_graph_type)
self._radio_button_graph_type.addButton(self._radio_button_oscillogram)
self._radio_button_graph_type.addButton(self._radio_button_spectrum)
graph_type_layout.addWidget(self._radio_button_oscillogram)
graph_type_layout.addWidget(self._radio_button_spectrum)
draw_parameters_layout.addLayout(graph_type_layout)

self._graph_type_stacked_layout = QStackedLayout()

self._oscillogram_control_widget = QWidget()
oscillogram_layout = QVBoxLayout()

self._oscillogram_control_start_input = QLineEdit()
self._oscillogram_control_end_input = QLineEdit()
self._oscillogram_control_start_label = QLabel("start")
self._oscillogram_control_end_label = QLabel("end")
start_layout = QHBoxLayout()
start_layout.addWidget(self._oscillogram_control_start_label)
start_layout.addWidget(self._oscillogram_control_start_input)
oscillogram_layout.addLayout(start_layout)
end_layout = QHBoxLayout()
end_layout.addWidget(self._oscillogram_control_end_label)
end_layout.addWidget(self._oscillogram_control_end_input)
oscillogram_layout.addLayout(end_layout)
self._oscillogram_control_widget.setLayout(oscillogram_layout)
self._graph_type_stacked_layout.addWidget(self._oscillogram_control_widget)
self._graph_type_stacked_layout.setCurrentIndex(0)
```

```
self._spectrum_control_widget = QWidget()
spectrum_layout = QVBoxLayout()
self._spectrum_control_widget.setLayout(spectrum_layout)
self._radio_spectrum_filter_limit_avg = QRadioButton("avg")
self._radio_spectrum_filter_limit_stdev = QRadioButton("stdev")
self._radio_spectrum_filter_limit_2stdev = QRadioButton("2x stdev")
self._radio_spectrum_filter_limit_3stdev = QRadioButton("3x stdev")
self._radio_spectrum_filter_limit_disabled = QRadioButton("disabled")
self._radio_spectrum_filter_limit_disabled.setChecked(True)
self._radio_spectrum_filter_group_limit = QButtonGroup()
self._radio_spectrum_filter_group_limit.addButton(self._radio_spectrum_filter_limit_avg)
self._radio_spectrum_filter_group_limit.addButton(self._radio_spectrum_filter_limit_stdev)
self._radio_spectrum_filter_group_limit.addButton(self._radio_spectrum_filter_limit_2stdev)
self._radio_spectrum_filter_group_limit.addButton(self._radio_spectrum_filter_limit_3stdev)
self._radio_spectrum_filter_group_limit.addButton(self._radio_spectrum_filter_limit_disabled)
spectrum_layout.addWidget(self._radio_spectrum_filter_limit_avg)
spectrum_layout.addWidget(self._radio_spectrum_filter_limit_stdev)
spectrum_layout.addWidget(self._radio_spectrum_filter_limit_2stdev)
spectrum_layout.addWidget(self._radio_spectrum_filter_limit_3stdev)
spectrum_layout.addWidget(self._radio_spectrum_filter_limit_disabled)
self._graph_type_stacked_layout.addWidget(self._spectrum_control_widget)

draw_parameters_layout.addLayout(self._graph_type_stacked_layout)

self._button_draw = QPushButton("display")
self._button_draw.clicked.connect(self.button_action_draw)
draw_parameters_layout.addWidget(self._button_draw)

main_layout.addLayout(draw_parameters_layout)

anomalies_detection_layout = QHBoxLayout()
self._anomalies_detection_label = QLabel("Anomalies:")
self._anomalies_detection_indicator = QLabel("")
self._listing_folder.itemClicked.connect(self.check_for_anomalies)
anomalies_detection_layout.addWidget(self._anomalies_detection_label)
anomalies_detection_layout.addWidget(self._anomalies_detection_indicator)
main_layout.addLayout(anomalies_detection_layout)
```

```
    self.setLayout(main_layout)


def check_for_anomalies(self):
    if model.is_anomalies_detected(self.gather_params()):
        self._anomalies_detection_indicator.setText("detected")
    else:
        self._anomalies_detection_indicator.setText("not detected")


def update_listings(self, filter_by: str):
    self._listings_stacked_layout.currentWidget().update_listing(filter_by)


def change_listing(self, button):
    button_to_id = {
        "Folder": 0,
        "Database": 1
    }
    self._listings_stacked_layout.setCurrentIndex(button_to_id[button.text()])
    self._listings_stacked_layout.currentWidget().update_listing(self._input_search.text())


def change_graph_type(self, button):
    button_to_id = {
        "oscillogram": 0,
        "spectrum": 1
    }
    self._graph_type_stacked_layout.setCurrentIndex(button_to_id[button.text()])


def button_action_draw(self):
    params = self.gather_params()
    plot(params)


def gather_params(self):
    return {
        "graph_type": self._radio_button_graph_type.checkedButton().text(),
        "start": self._oscillogram_control_start_input.text(),
        "end": self._oscillogram_control_end_input.text(),
        "p": self._checkbox_dimension_p.isChecked(),
```

```
            "vx": self._checkbox_dimension_vx.isChecked(),
            "vy": self._checkbox_dimension_vy.isChecked(),
            "vz": self._checkbox_dimension_vz.isChecked(),
            "spectrum_delimiter": self._radio_spectrum_filter_group_limit.checkedButton().text(),
            "location": self._radio_button_group.checkedButton().text(),
            "label": self._listings_stacked_layout.currentWidget().currentItem().text()
            if self._listings_stacked_layout.currentWidget().selectedItems() else None
}
```

ДОДАТОК 3

Програмні засоби інтелектуально-статистичної обробки гідроакустичних сигналів

Опис програми

УКР.НТУУ”КПІ ім. Ігоря Сікорського”_ТЕФ_АПЕПС_ТМ62202_20Б 13-1

Аркушів 8

Київ – 2020

АНОТАЦІЯ

Програмна система надає можливість виконувати інтелектуально-статистичну обробку гідроакустичних сигналів. В програмному продукті реалізовано комп’ютерний графічний користувацький інтерфейс для легкої взаємодії з нею. Використовуючи програму, користувач може обробити певний файл або запис гідроакустичного сигналу в базі даних. Після обробки з’являється можливість виконувати візуалізацію сигналу а також отримати інформацію про присутність в ньому аномалій.

Застосунок створено в середовищі розробки JetBrains PyCharm 2020 з використанням мови програмування Python для запуску основного вікна програми, бази даних PostgreSQL та контейнерів Docker для легкого розгортання.

ЗМІСТ

-4-

ЗАГАЛЬНІ ВІДОМОСТІ

Створений програмний продукт складається з головного керуючого вікна Python, бази даних PostgreSQL, системи контейнеризації Docker.
Вікно керування програмою запускається в головному Python процесі, сервер керування базою даних знаходиться в контейнері. За умови надання правильної конфігурації, можна налаштувати запис зовні контейнеру, для збереження даних, після його знищення. Кожне вікно візуалізації виконується в окремому Python процесі, який зупиняється та знищується після закриття вікна. Програмна система надає можливість одночасного відкриття багатьох вікон візуалізації.
Система надає можливість здійснювати інтелектуально-статистичну обробку гідроакустичних сигналів, та відображати їх у виді осцилограми або спектру.

-5-

ФУНКЦІОНАЛЬНЕ ПРИЗНАЧЕННЯ

Розроблена програмна система створена для інтелектуально-статистичної обробки гідроакустичних сигналів. Вона може бути використана для виконання дослідження сигналів на наявність аномалій та візуалізації цих сигналів.

-6-

ОПИС ЛОГІЧНОЇ СТРУКТУРИ

Розроблена програмна система складається з декількох компонентів:

— Основний додаток, що виконується в головному процесі Python;

— Вікна візуалізації даних, що виконуються в побічних процесах Python, процеси запускаються при відкритті вікна візуалізації певного сигналу та зникають після закриття цього вікна, може бути одночасно відкрито декілька вікон візуалізації

— Контейнер з базою даних PostgreSQL всередині, для легкого розгортання системи

— Мережа для взаємодії з контейнерами Docker-Compose

Взаємодія всіх вказаних вище компонентів між собою надає можливість коректного виконання програми.

-7-

ВИКОРИСТОВУВАНІ ТЕХНІЧНІ ЗАСОБИ

Для використання програмного продукту необхідний комп’ютер з підтримкою віртуалізації, достатні ресурси для розгортання контейнерів та запуску одночасно декількох процесів мови Python та наявність інтерпретатору мови Python.

-8-

ВХІДНІ І ВИХІДНІ ДАНІ

Вхідними даними можуть виступати файли з специфічним форматом (за замовчуванням, це файли що складаються з пустого заголовку розміром в 4 байти та блоків даних по 4 значення розміром в 2 байти, ці дані повинні представляти собою гідроакустичний сигнал з частотою дискретизації 2048Гц), який вказаний в конфігурації проекту.

Вихідні дані представляються собою текстову та графічну інформацію про гідроакустичний сигнал.

ДОДАТОК 4

Програмні засоби інтелектуально-статистичної обробки гідроакустичних сигналів

АПРОБАЦІЯ

Матеріали VII Науково-практичної дистанційної конференції фахівців з розробки програмного забезпечення (VII Scientific and Practical Virtual Conference of Software Development Specialists) 2020 року,

м. Київ, КПІ ім. Ігоря Сікорського, 1 липня 2020 року

УКР.НТУУ”КПІ ім. Ігоря Сікорського”_ТЕФ_АПЕПС_ТМ62202_20Б

Аркушів 8

2020

Ministry of Education and Science of Ukraine

National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute"

Glushkov Institute of Cybernetics, National Academy of Sciences of Ukraine

Polytechnic Institute of Tomar (Portugal)

Smart Cities Research Center (Portugal)

MODERN ASPECTS OF SOFTWARE DEVELOPMENT

Proceedings of
VII Scientific and Practical Virtual Conference of
Software Development Specialists

June, 1, 2020

Kyiv, Ukraine

CONTENTS

Mykhal'ko V.H., Student,
Kublii L.I., Candidate of Sciences in Technology,
National Technical University of Ukraine
"Igor Sikorsky Kyiv Polytechnic Institute"
Ukraine, Kyiv, street Polytechnic, 6,
tel. 050-713-50-61, 063-719-12-31,
e-mail: mykhalko.edu@gmail.com, kublii_l_i@ukr.net

Software tools for intellectual and statistical processing of hydroacoustic signals

Hydroacoustic studies the phenomena that occurs in the aquatic environments and associated with the formation, propagation and reception of acoustic waves [1]. But despite the large number of sonar devices, hydroacoustic still faces big problems with data accuracy.

Collected from sonar devices data can be processed and analyzed in different ways. One of the promising areas is usage of machine learning [2]. Creation of artificial conditions is a great possibility to generate required amount of digital information. With this information it will be possible to create machine learning models that will be used for further data forecasting or anomalies detection in aquatic environments.

With statistical processing and modeling, you can get more accurate data about the location, size, speed, nature of objects. Machine learning can also provide opportunity to create systems for detection and elimination of extraneous noise in data obtained from sonar devices. That kind of information can be used in the military, maritime navigation, fish exploration, oceanographic research, etc.

The system for processing sonar signals is implemented using Python programming language [3]. The analysis of sonar signals is primarily reduced to the processing of information from audio tracks, presented in .wav format. Files of this format have a conform structure that includes meta-information part and the audio data itself. This structure is shown in table 1.

Table 1. Audio track data structure

Cell size	Cell name	Description
4	Chunk ID	“RIFF” chunk descriptor. The Format concerns
4	Chunk size	
4	Format	
4	Subchunk1 ID	“fmt” sub-chunk
4	Subchunk1 size	
2	Audio format	
2	Num channels	
4	Sample rate	
4	Byte rate	
2	Block align	
2	Bits per sample	
4	Subchunk2 ID	“data” sub-chunk
4	Subchunk2 size	
*	Data	

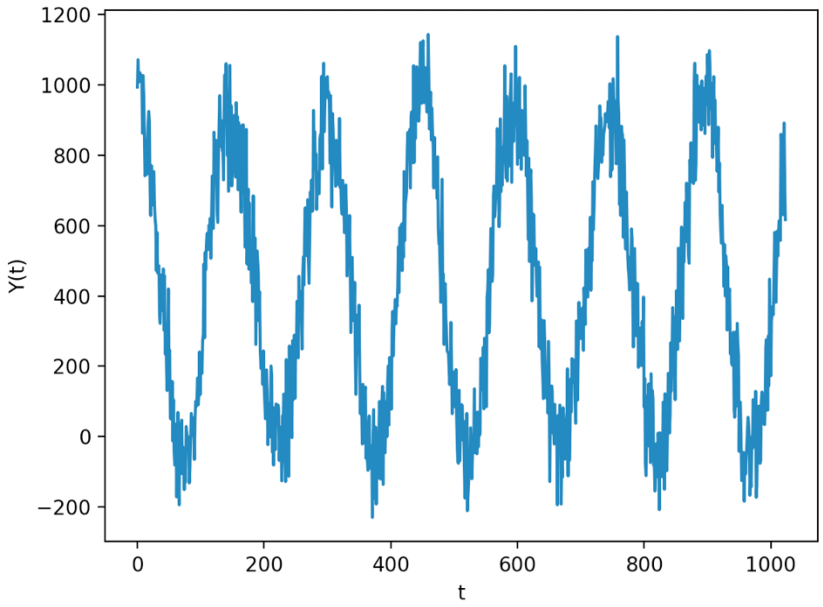
Initial step of dataset creation is processing of audio track’s meta-information. Then, based from information received, it becomes possible to read raw audio data, recorded in the .wav-file. This step can be simplified by usage of Python package called wave, it contains predefined functions for .wav-files processing, but in that case some additional flexibility in data-processing can be lost. If usage of this library does not meet certain requirements, the built-in Python struct package can be used, which allows iteratively walk through the data stream of the file and read the bytes in the specified order.

One more approach is usage of ready-made dataset created by the special sensors. Those data will be partially similar to the data, gathered from audio tracks, but may contain more accurate and additional values. The special case of extended data is data that consists

not only from the values of the pressure at a certain point (p), but also the displacement of the particles of the medium along the coordinate axes (v_x, v_y, v_z). Such data is more accurate due to the possibility to work with the same signal in several dimensions. Filtration, noise reduction, values extraction, etc. will be much more efficient with that kind of data.

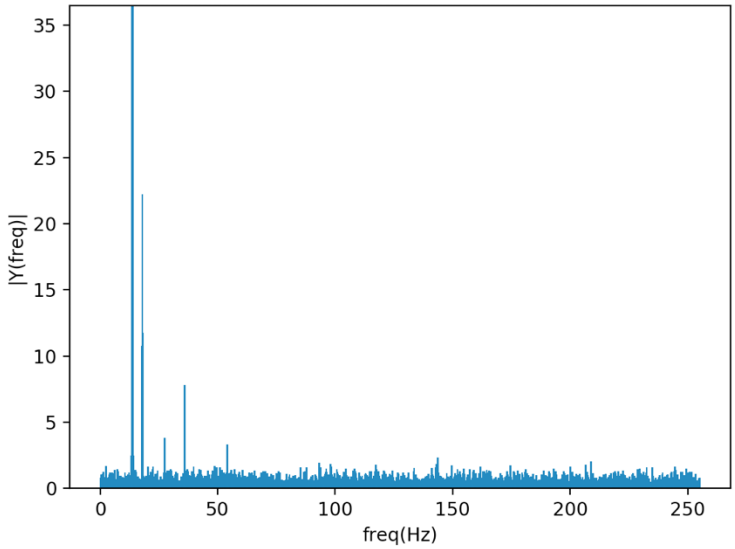
For forecasting execution, samples of relevant data (a set of data that is relevant to audio signals and provides metrics, that needs to be predicted) required along with audio data. Anomalies detection does not require any additional information, this method focuses on searching of instabilities in the input data.

From the signal data obtained, it is possible to build an oscillogram, which can visually, to some extent, show some information about the signal, but that kind of data will not be effective on machine learning model creation without pre-processing. An example of an oscillogram is given on picture 1.



Picture 1 – Oscillogram for 0-0.5 s.

In the case of signal analysis, the spectrum will be effective for analysis. Since sonar signals frequencies are quite low, it makes sense to limit them to range of [0; 255]. Picture 2 shows the spectrum of the same signal was given on the oscillogram.

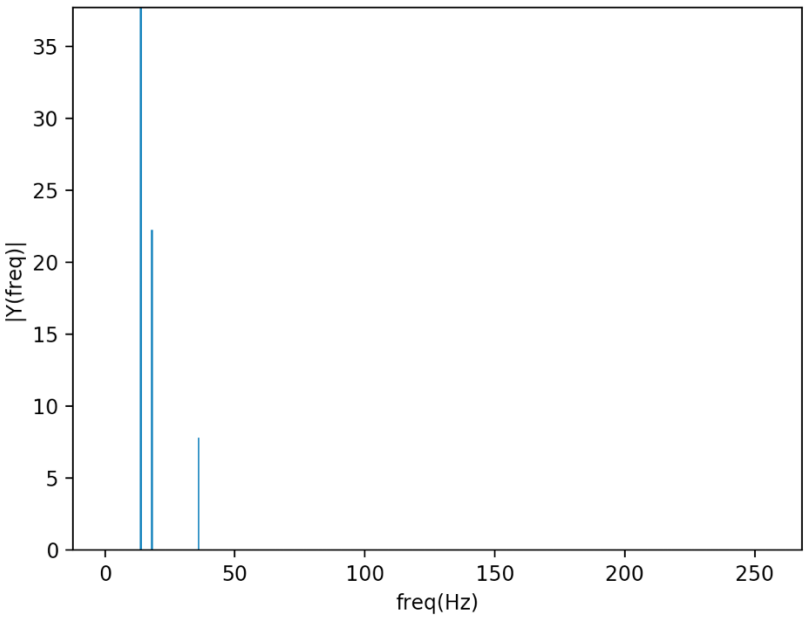


Picture 2 – Signal spectrum.

Signal, described by spectrum, will be much more efficient for model creation. Since sonar signal spectrum can is limited by specified range, we can send those data to machine learning model as vector, sized by length of specified range, in our case size equals to 256. But still data efficiency can be increased by noise reduction. One of the possible solutions to reduce noises is usage of standard deviation, but in reverse notation: abnormal values should be kept, normal values should be filtered out. Picture 3 shows example of signal spectrum with filtered noises.

After filtration execution, the data will become effective for creation of machine learning model.

Information from audio tracks is translated into a tabular view and stored within the database which is related to the implemented system. The database also stores project configuration and output data. The PostgreSQL relational system was chosen as the database management system; program interacts with database by usage of python package called psycopg2 in combination with ORM-component SQLAlchemy. MatPlotLib is used to display information from audio tracks and resulting graphs. Thanks to the visual presentation of information, you can easily see the relationship between input and output data.



Picture 3 – Filtered signal spectrum.

User interface is created using PyQt5 library modules. Data processing and calculation are performed by the functions of libraries NumPy (work with multidimensional arrays), SciPy (processing of numerical sequences), Pandas (work with two-dimensional tables). The scikit-learn machine learning library was used to build the models. The project code configures the interaction between the selected packages and connects all the components into a single system.

Implemented software product can provide information about objects and detect anomalies in the aquatic environment.

List of references

1. Румынская И.А. Основы гидроакустики. — Л.: Судостроение, 1979. — 209 с.

2. Вьюгин В.В. Математические основы машинного обучения и прогнозирования. — М.: МЦНМО, 2018. — 384 с.

3. Python 3.8.3 documentation [Electronic resource]. — Access mode: https://docs.python.org/3/